Calhoun: The NPS Institutional Archive

DSpace Repository

Theses and Dissertations 1. Thesis and Dissertation Collection, all items

1981

A computational procedure for the protection of industrial power systems.

Salmond, Charles N.

Pennsylvania State University

http://hdl.handle.net/10945/20461

Downloaded from NPS Archive: Calhoun

Calhoun is the Naval Postgraduate School's public access digital repository for research materials and institutional publications created by the NPS community. Calhoun is named for Professor of Mathematics Guy K. Calhoun, NPS's first appointed -- and published -- scholarly author.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

http://www.nps.edu/library

terey, CA 939

T21153

Approved for public release;
distribution unlimited.

The Pennsylvania State University

The Graduate School

Department of Electrical Engineering

A Computational Procedure for the Protection of Industrial Power Systems

A Thesis in

Electrical Engineering

by

Charles N. Salmond

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

November 1981

I grant The Pennsylvania State University the nonexclusive right to use this work for the University's own purposes and to make single copies of the work available to the public on a not-for-profit basis if copies are not otherwise available.

Charles N. Salmond

31545
c. 1

ABSTRACT

A procedure is presented that allows an engineer to coordinate protective devices consisting of fuses, circuit breakers, and overcurrent relays based upon load and fault currents. The resulting digital-computer program makes many of the decisions requiring very little prior in-depth knowledge of protective devices and power systems for its utilization.

The coordination procedure consists of six separate programs that must run in sequence, as the output from one program forms the input for the next program. This reduces in size of internal computer storage required.

The theory is presented with an explanation of each algorithm followed by a basic flow chart. An example computer run is presented at the end.

TABLE OF CONTENTS

Page

ABSTRACT iii

LIST OF TABLES vi

LIST OF FIGURES vii

ACKNOWLEDGMENTS viii

1. INTRODUCTION 1
   Problem Statement 2

2. CURVE SMOOTHING AND DIGITIZING 8
   General 8
   Curve Smoothing 8
   Summary 14

3. THE INPUT OF DISTRIBUTION-SYSTEM DATA 15
   General 15
   Per Unit Reduction 15
   Formulae for Line Resistance and Reactance 18
   Summary 20

4. LOAD-FLOW ANALYSIS 23
   General 23
   Load Flow 23
   Summary 27

5. FORMATION OF IMPEDANCE-BUS NETWORK 30
   General 30
   Formation of Z-bus 30
   Summary 38

6. FAULT CALCULATIONS 39
   General 39
   Summary 42

7. PROTECTIVE-DEVICE COORDINATION 44
   General 44
   Device Coordination 44
   Fuse Selection and Coordination 48
   Relay Selection and Coordination 49
   Molded-Case Circuit Breaker 51
   Summary 54

Page

8. OPERATING THE PROGRAMS 59
General 59
One-Line Diagram 59
Load Flow 61
Solution of Fault Currents 61
Device Coordination and Plotting 62
Summary 79

9. CONCLUSIONS 80
Recommendations for Further Work 81

REFERENCES 83

APPENDIX A
CURVE-SMOOTHING PROGRAMS 85

APPENDIX B
COORDINATION AND PLOTTING PROGRAMS 90

APPENDIX C
DISK-FILE FORMAT 151

APPENDIX D
SAMPLE PROGRAM DATA INPUT 154

LIST OF TABLES

Table Page

3.1. Variable definitions 16

5.1. Equations for the formation of ZBUS 33

7.1. Fuse-selection coefficients 50

7.2. Relay-curve coefficients 52

7.3. Molded-case circuit breaker 55

8.1. Overcurrent-relay types 64

8.2. Current-limiting fuses 66

8.3. Boric fuses 67

8.4. Molded-case circuit breakers 68

8.5. Disc files 69

8.6. Mine power cable polynomial coefficients 70

LIST OF FIGURES

Figure Page

1.1. Flow chart of the coordination procedure 4

2.1. Curve-digitizing flow chart 12

3.1. Per-unit calculations flow chart 21

4.1. Newton-Raphson load-flow chart 29

5.1. Transformation of square sparse primitive impedance 31

5.2. Zbus^{012} formation flow chart 35

6.1. Flow chart for FALT.F4 43

7.1. Two possibilities for coordination element Z 46

7.2. Coordination flow chart 56

7.3. Device-graphing flow chart 58

8.1. One-line diagram of sample system 60

8.2. Example coordination plot 75

ACKNOWLEDGMENTS

The author is indebted to Dr. Fredrick Trutt and Dr. Lloyd A. Morley for their support and guidance throughout this thesis project.

Dr. William S. Adams and his staff of technicians in the Electrical Engineering Hybrid Computer Laboratory, Mr. John Griebling, and Mr. Robert Hirlinger were especially helpful.

The author appreciated the comments and suggestions of Dr. George A. Etzweiler in his critical review of this work.

1. INTRODUCTION

A prime objective of electrical engineers working on the design, operation, or maintenance of electrical power systems is to provide reliable, efficient power to their customers. The distribution system has always been a weak link in the power chain. Often, the catastrophic results of a fault are easily seen, but the cause may be extremely difficult to find. A distribution system, either in the utility or the user's facilities that is not adequately protected from electrical faults is both dangerous and costly. An unsatisfactorily protected system can cause considerable inconvenience and damage to equipment. Adequate protection is not a set definition. Each load system is different with its own set of boundary values.

In a distribution system, the flexibility of design reduces near the generator or the load. Usually the generator or load protection is designed as an integral part of the equipment by the manufacturer and any changes should only be undertaken with their counsel. Moving away from the generator or load, the priority shifts from load protection to distribution protection. It must never be forgotten that the load, distribution, and generation systems must be protected in such a way that a device that operates satisfactorily in one system cannot have a detrimental effect on another system. In this thesis the distribution section is attached from an infinite-bus power source to the load-protection device. The programs provide the

proper setting of a chosen device in order to provide good protective coordination. Proper coordination is achieved when a device closest to the fault clears the fault or overload condition before the rest of the system is disturbed. In order to achieve this end by manual means, extensive mathematical calculations and experience are required. As a direct result, proper coordination is often not achieved. The use of digital computers to solve the coordination problem makes a more exacting solution. However, computers with large internal memory are required to solve these problems. Several approaches to computer solutions have been published [1-4].

Problem Statement

The purpose of this thesis is to develop the procedure and an interactive computer program for a generalized case of device coordination. This coordination should include fuses, relays, and molded case circuit breakers. The phases of research are as follows.

1. Develop a computer procedure to compute the one-line program per unit values.

2. Develop a computer procedure to perform a load-flow analysis.

3. Develop a computer procedure to compute line-to-line ground and three-phase faults.

4. Develop and apply procedures for mathematical representation of line X/R data, fuse, molded-case circuit breaker, and relay-curve time-current characteristics.

5. Develop a computer procedure to coordinate the devices from phase 4 with the data derived from phases 2 and 3.

Although the coordination programs are written for the PDP10 computer, it requires no more than 25K bytes of internal memory and a disk unit to solve any system of 30 buses and 100 elements. Almost any large system can be reduced to its smaller parts composed of 30 buses or less. As each segment is coordinated, it can be represented by a load and its protective device. In this manner a 300-bus system could be divided into smaller segments and in 11 runs would be completely coordinated. The step-by-step approach used here allows for insertion or extraction of data before proceeding to the next step. Figure 1.1 is a basic flow chart of the coordination program. Hereafter, it will be referred to by its individual parts.

When the coordination process has been completed, a load flow study and a fault analysis has been performed as well. The major assumptions that have been made are listed here. Other minor ones are included in the corresponding chapters. These assumptions have been made in an effort to keep the solution as general as possible.

1. The incoming line is assumed to be an infinite bus.

2. Assymetrical currents are neglected. Devices are coordinated on the basis of symmetrical fault currents.

3. Only synchronous-motor contributions to the fault currents are recognized. If a large induction motor

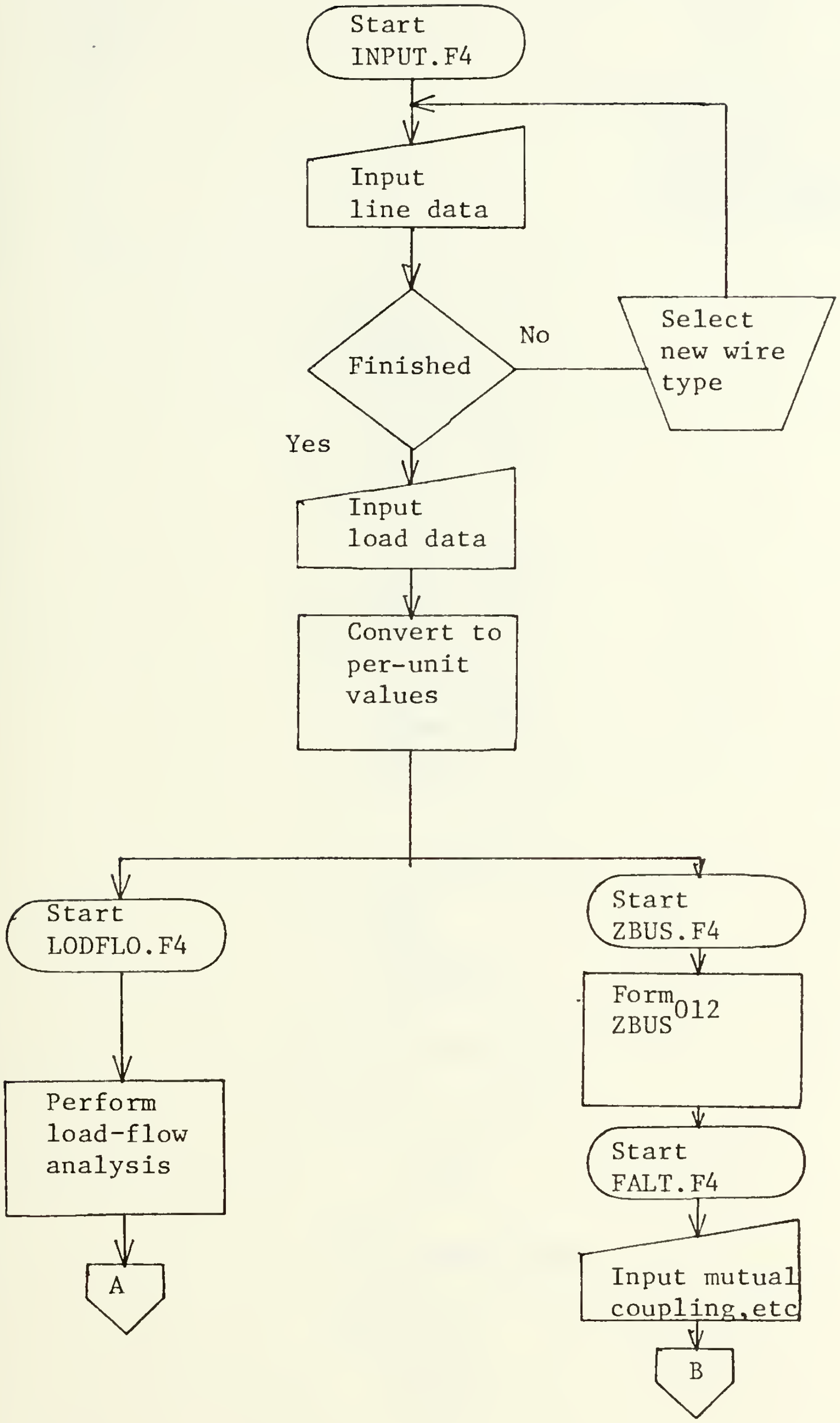


Figure 1.1. Flow chart of the coordination procedure.

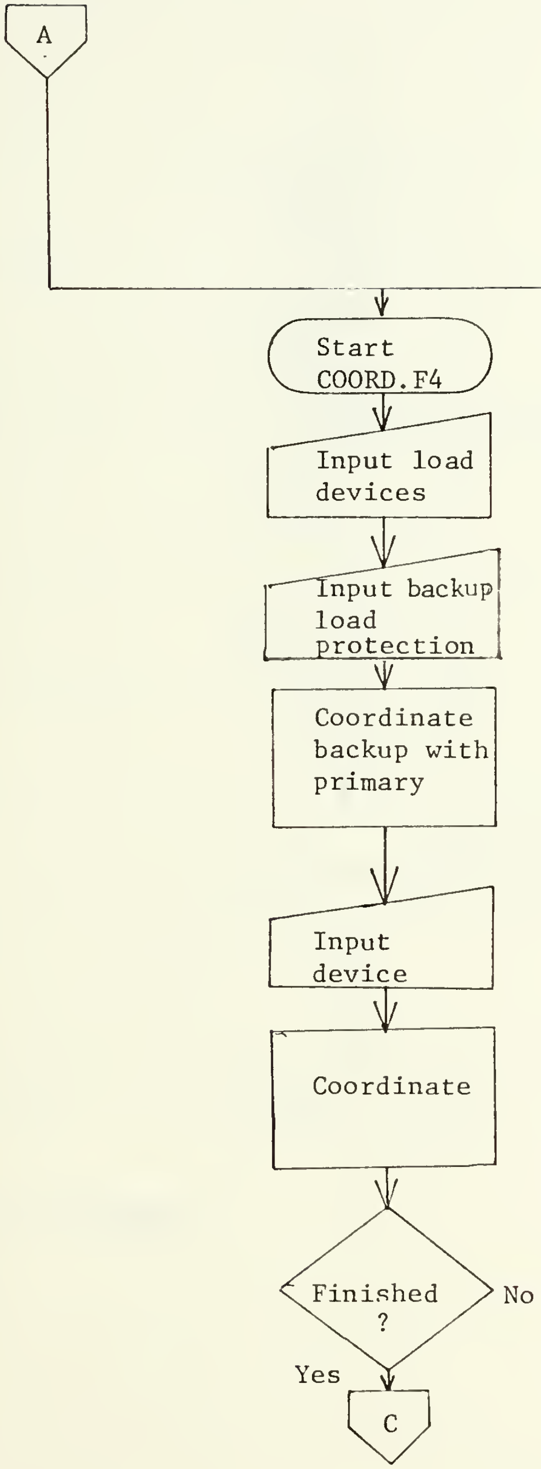

A
B
Solve for 3ϕ, line to ground faults
Start COORD.F4
Input load devices
Input backup load protection
Coordinate backup with primary
Input device
Coordinate
Finished ?
No
Yes
C

Figure 1.1. Continued.

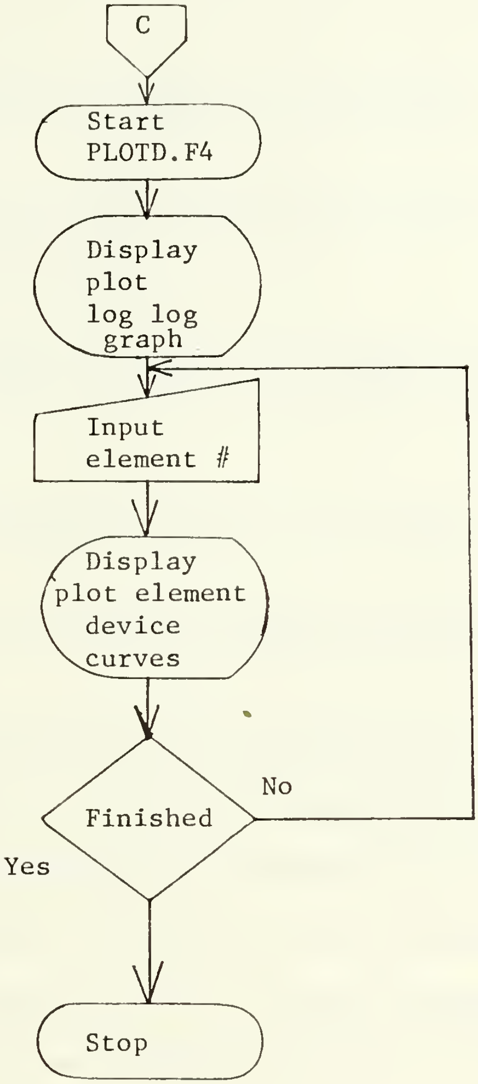


Figure 1.1. Continued.

is present, it may be added as a fictitious synchronous motor.

4. Line-charging shunt admittance is ignored.

5. Prefault current and voltages are neglected in fault-current computations.

6. Different sizes of fuses have the same time-current curve shape.

7. Overload conditions are 125% of full load.

8. The system has a balanced three-phase load.

In this thesis each phase of research in the problem statement has a chapter devoted to it. After describing the computational methods used in curve smoothing, the coordination problem begins with data input. Next the fault currents are solved followed by the actual coordination program and device response graphing. Finally, a sample distribution system coordination problem is described.

2. CURVE SMOOTHING AND DIGITIZING

General

In this chapter, the methods for using a data digitizing "numonics" device and the curve smoothing program of Appendix A are explained. The various device equations that were found that adequately represented the particular device's time current curve are also described.

A method was described in Wagner [1] for performing this by using the PDP10 computer. This least-square curve-fitting program [1] was expanded to eliminate any external subroutines that may not be available. The coordination program makes extensive use of polynomial functions whose coefficients were derived by using the least-squares curve-smoothing technique.

Curve Smoothing

Electrical Cables. Coefficients for the mathematical modeling of resistance and reactance were taken from data for mine power cable [5]. This is used as only one type of cable or wire. Any other type may be modeled by the same procedure. For instance, either actual cable or line data could be used directly [6], or the theoretical resistance and reactance for each size could be calculated [5-8].

Fuses. Two types of fuses are used in the coordination program. These are current-limiting and solid-material boric-acid power fuses. In each case, the curve shape is assumed to be the same for different sizes. The low-current asymptotic value is called the "ofset." Each fuse has a minimum and a typical operating time. In fuse coordination, both values must be used in describing the operating envelope of a particular fuse. The higher typical operating time curve is referred to as "Ofset 1." By this means, any fuse can be completely described by four equations, where:

$$\text{Log } T_{min} = (\sum_{j=o}^{n} a_j (\text{Log Current} - \text{Ofset})^j - 2 \qquad (2.1)$$

$$\text{Log } T_{yp} = (\sum_{k=0}^{n} a_k (\text{Log Current} - \text{Ofset1})^k - 2 \qquad (2.2)$$

Fuses are selected also by using mathematical models. To select a fuse, the program uses the overload-current value as the variable in the polynomial function. The result is the device number. In curve smoothing, variables can assume no value less or greater than those originally used as data for the curve-smoothing program, or large errors can result.

Molded-Case Circuit Breakers. Like the fuses discussed above, each molded-case circuit breaker consists of an operating envelope bounded by a minimum value and a maximum value. Unlike fuses, however,

a table look-up approach is used in selection of a device and its setting because of the smaller number of sizes and magnetic-trip settings.

Relays. Wagner [1] discusses the representation of the family of relay curves for each type with one polynomial equation comprised of four coefficients. The 0th power coefficient was left out as a variable for coordination. Unfortunately, this method proves unsatisfactory when working with more than one relay type. It was found that each relay could be represented by five coefficients. The 0th power coefficient is necessary in that it provides the time displacement necessary for minimum time dial setting. The curves were derived using the procedure below.

Westinghouse Company type relays were used. An average curve was taken as time-dial 4. This curve was reduced to a polynomial function. The time difference between time-dial 4 and time-dial 1/2 taken at 20 times the pickup value is subtracted from the 0th coefficient. Twenty times pickup was a good value for computing the time difference as the time value changes very little for increases in current beyond that point. The resultant curve was the average shape of a time-dial setting of 4 now shifted to minimum setting. This allows for time values computed during coordination to be summed with the 0th coefficient. The curves are then represented as follows:

$$\text{Log Time} = \left(\overset{n}{\underset{j=0}{}} \; a_j (\log(\text{Current Value})^j) \right) +$$

$$\text{Log (Time Setting)}$$

This allows the use of more than one type of relay in each coordination as it separates time setting value from relay type.

Curve Digitizing. The device's time versus current curves were digitized by using a NUMONICS electronic graphics calculator (EGC). The NUMONICS EGC is designed to translate graphic information on any medium into digital information. Complete information on its usage may be found in -NUMONICS Electronic Graphics Calculator [9]. The NUMONICS EGC then provided a convenient way to enter X-Y coordinate data representing any curve onto disk. Since this device works in length in inches, the data has to be normalized to its proper units.

A simple program DIGIT.F4 was written to normalize and store onto disk the NUMONICS EGC information. The program flow chart is shown in Figure 2.1.

Curve Smoothing. The curve-smoothing program contained in Wagner [1] was used as a basis for the curve smoothing program DEV.F4. Unnecessary parts were removed and a double precision matrix inversion routine was added. This prevented the necessity of using subroutines contained in the computer's SSP routine files. The inversion routine was converted from The Pennsylvania State University Computer Center's matrix inversion routine CMINV which inverts a complex square matrix. This routine in turn was converted from the SSP routine MINV [10].

Since it has its own inversion routine, this curve smoothing program can be easily expanded to any number of points and any dimension of fit by changing the dimension statements. The flow chart and procedure remains the same as in [1]. A complete program listing may be found in Appendix A.

D
N

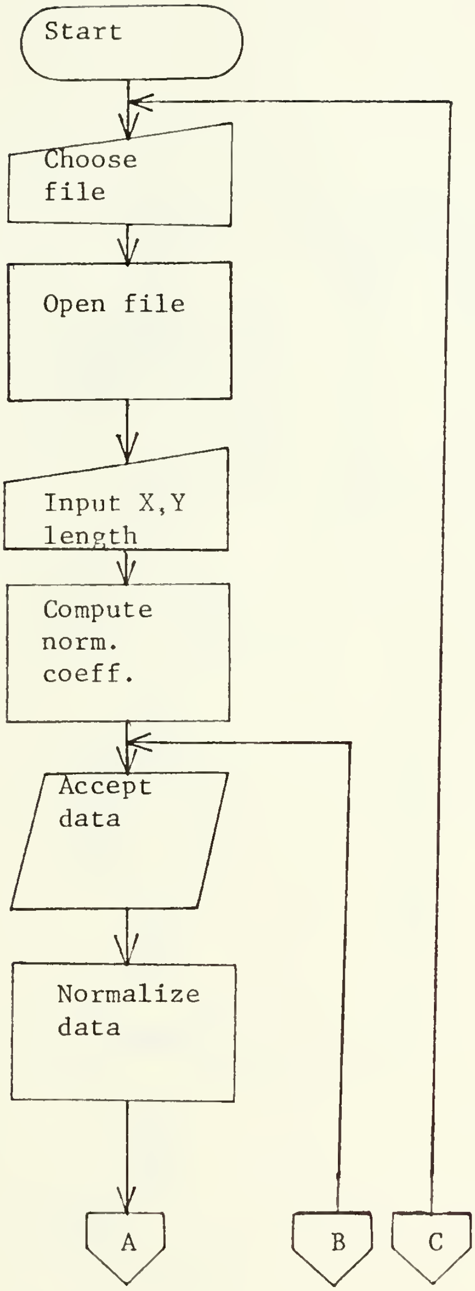


Figure 2.1. Curve-digitizing flow chart.

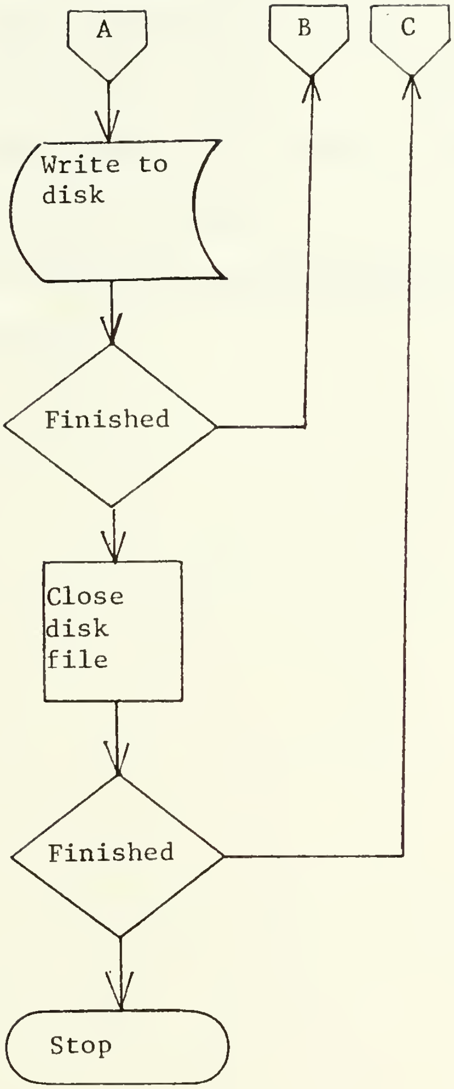


Figure 2.1. Continued.

Summary

Polynomial functions describe the various device-response curves, line X/R data, and fuse-selection criteria. The coefficients for these polynomial functions are found from data entered using a digital-data device or manual input. This data is reduced by a curve-smoothing program to the desired function.

The start of the coordination problem begins in the next chapter with line and load data entry.

3. THE INPUT OF DISTRIBUTION-SYSTEM DATA

General

Usually, the first step in performing any calculations on a distribution system is to reduce all system values to per unit. The methods used to do this in the program will be described in this chapter. Errors can be introduced easily when converting to the per unit system using manual calculations. Program INPUT.F4 was written to perform this function as well as computing line resistance and reactance values and storing all system data on disk for use in subsequent programs.

Per-Unit Reduction

The program uses the following formulas to convert to the per-unit system [8]. (See Table 3.1 for variable definitions used in this thesis.)

$$Z_{\text{Base}} = (KV_{\text{Base}})^2 / MVA_{\text{Base}}$$

$$Z_{\text{P.U.}} = Z(\Omega) / Z_{\text{Base}}$$

$$I_{\text{Base}} = KVA_{\text{Base}} / KV_{\text{Base}}$$

Table 3.1. Variable definitions.

$Z(\Omega)$	element impedance
$Z_{P.U.}$	impedance in per unit
X_d	subtransient reactance
P_p	real power bus p
Q_p	reactive power bus p
E_p	complex voltage bus p
I_p	complex current bus p
Y_{pq}	Y bus element bus p to q
$\bar{y}_{pq,rs}$	admittance value from primitive admittance matrix
Z_{pq}	Z bus element bus p to q
$z_{pq,pq}$	impedance value from primitive impedance matrix
z_f	fault impedance
y_f	fault admittance

$$Z_{P.U.new} = Z_{P.U.given} \left(\frac{KV_{Base\ given}}{KV_{Base\ new}}\right)^2 \left(\frac{KVA_{Base\ new}}{KVA_{Base\ given}}\right)$$

Utilizing the above formulas it is possible then to represent all elements in the distribution system on a per-unit basis. Loads need to be included also in the one-line representation.

The program allows loads to be represented either by induction motor horsepower, synchronous motor horsepower, kVA, or current.

Loads are entered in the following format: Bus number/horsepower, kVA, current/power factor. The program assumes 100% efficiency so one must use:

$$HP_{Entered} = HP/Efficiency \qquad (3.1)$$

If information on efficiency and power factors are unknown, the following equations may be used instead of 3.1 [9].

For induction motors, use:

$$HP_{Entered} = HP \times 1.1627$$

$$\text{Power Factor} = .86 \qquad (3.2)$$

For synchronous motors use:

$$\text{Unity Power Factor HP} = 1.139 \times \text{Horsepower} \qquad (3.5)$$

$$.8\ \text{Power Factor HP} = 1.18 \times \text{Horsepower} \qquad (3.6)$$

$$Z_{P.U.new} = Z_{P.U.given} \left(\frac{KV_{Base\ given}}{KV_{Base\ new}}\right)^2 \left(\frac{KVA_{Base\ new}}{KVA_{Base\ given}}\right)$$

Utilizing the above formulas it is possible then to represent all elements in the distribution system on a per-unit basis. Loads need to be included also in the one-line representation.

The program allows loads to be represented either by induction motor horsepower, synchronous motor horsepower, kVA, or current.

Loads are entered in the following format: Bus number/horsepower, kVA, current/power factor. The program assumes 100% efficiency so one must use:

$$HP_{Entered} = HP/Efficiency \tag{3.1}$$

If information on efficiency and power factors are unknown, the following equations may be used instead of 3.1 [9].

For induction motors, use:

$$HP_{Entered} = HP \times 1.1627$$

$$\text{Power Factor} = .86 \tag{3.2}$$

For synchronous motors use:

$$\text{Unity Power Factor HP} = 1.139 \times \text{Horsepower} \tag{3.5}$$

$$.8 \text{ Power Factor HP} = 1.18 \times \text{Horsepower} \tag{3.6}$$

Formulae for Line Resistance and Reactance

Element data is entered in the following format: line bus/ load bus/cable length in feet/wire size/transformer P.U. impedance/ size in MVA/line voltage in KV/load voltage in KV.

Wire sizes that are entered in American wire gauge (AWG) are converted to MCM wire gauge by using the following formula [1]:

$$\text{MCM} = [(.005)[(1.229)^{(36\ \text{AWG})}]^{2} \times 1000 \qquad (3.7)$$

Once the wire size is in MCM, the line resistance and reactance are computed using the polynomial coefficients that were found by the procedure in Chapter 2.

$$\text{Resistance or reactance} = \sum_{j=0}^{n} a_j [\log(\text{MCM})]^{j} \qquad (3.8)$$

The program should not be limited to just one wire type. The program does this by reading wire coefficients a_j above from disk so that each time the program is run it can compute element data using different wire types until the entire system has been completely entered.

Transformers become very important in that they can become the highest source of resistance and reactance. By their connections and neutral impedance, they determine the magnitude of zero-sequence current that will flow in the particular element. The program allows for this input and writes this information in the particular disk files for use later on in fault analysis.

Transformer inrush and withstand values must also be calculated. These values must be used in coordinating protective devices. These devices must pass inrush currents but open under conditions of exceeded transformer withstand currents. The algorithm in the program is based on average values. If test data is available this should be entered directly by changing the transformer data disk file directly after the program is run and before proceeding with coordinations. Inrush and withstand values are computed as follows [11]:

$$\text{Inrush} = 12\text{x}(\text{Transformer Base Current}) \tag{3.9}$$

$$\text{Withstand @2 Sec} = 25\text{x}\ (\text{Transformer Base Current}) \tag{3.10}$$

$$\text{Withstand @5 Sec} = 14.5\text{x}\ (\text{Transformer Base Current}) \tag{3.11}$$

Transformer withstand is some value between the two second value to the five second value and depends upon the transformer impedance. It is fairly easy, however, to have the device protect the complete range of withstand values and the program is written to accomplish this task.

Synchronous subtransient reactance is computed as follows [11]:

$$(\text{Per unit KVA})\ \text{x}\ (.17) = jX_d \tag{3.12}$$

One of the problems encountered in computing transmission line impedance is a reasonable calculation of the zero-sequence impedance. The computer program assumes that this value is two times the positive-sequence impedance [8] if nothing is entered when the computer asks for the zero-sequence impedance multiple. If a different value is entered, the computer multiplies the positive sequence reactive value by this multiple to arrive at the zero-sequence value.

A basic flow chart follows in Figure 3.1 with a complete program listing in Appendix B.

Summary

The first step toward solving the coordination problem is completed. All element data has been converted to the per-unit representation. The next logical step is to perform a load-flow analysis to determine the full load currents in each element.

Flow Chart Input

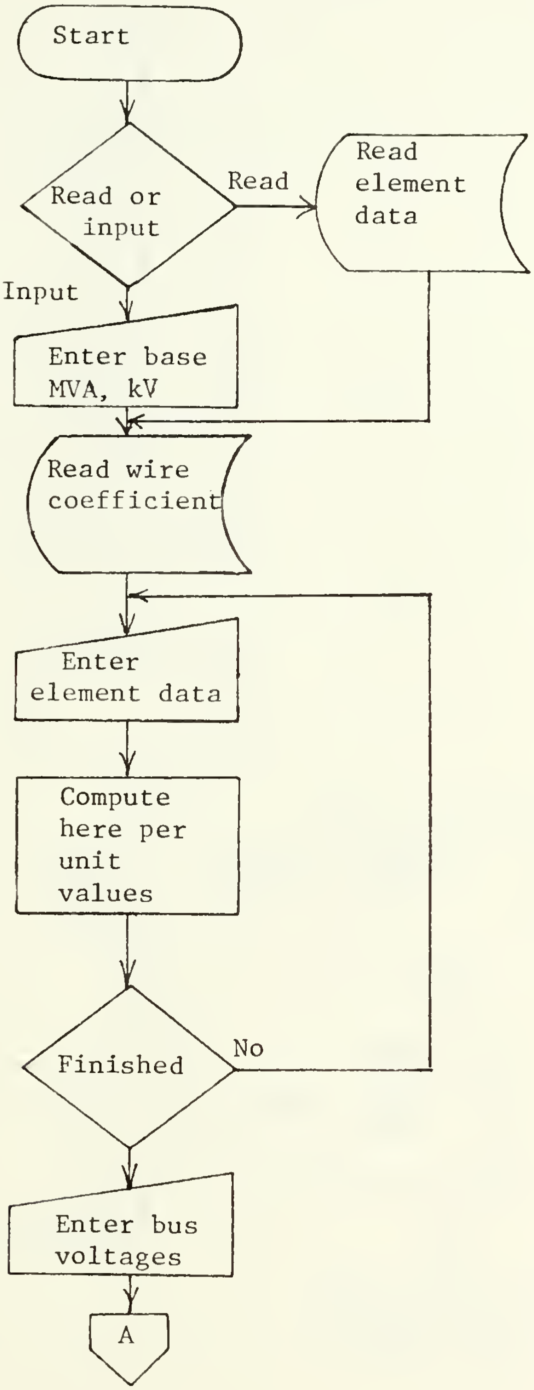


Figure 3.1. Per-unit calculations flow chart.

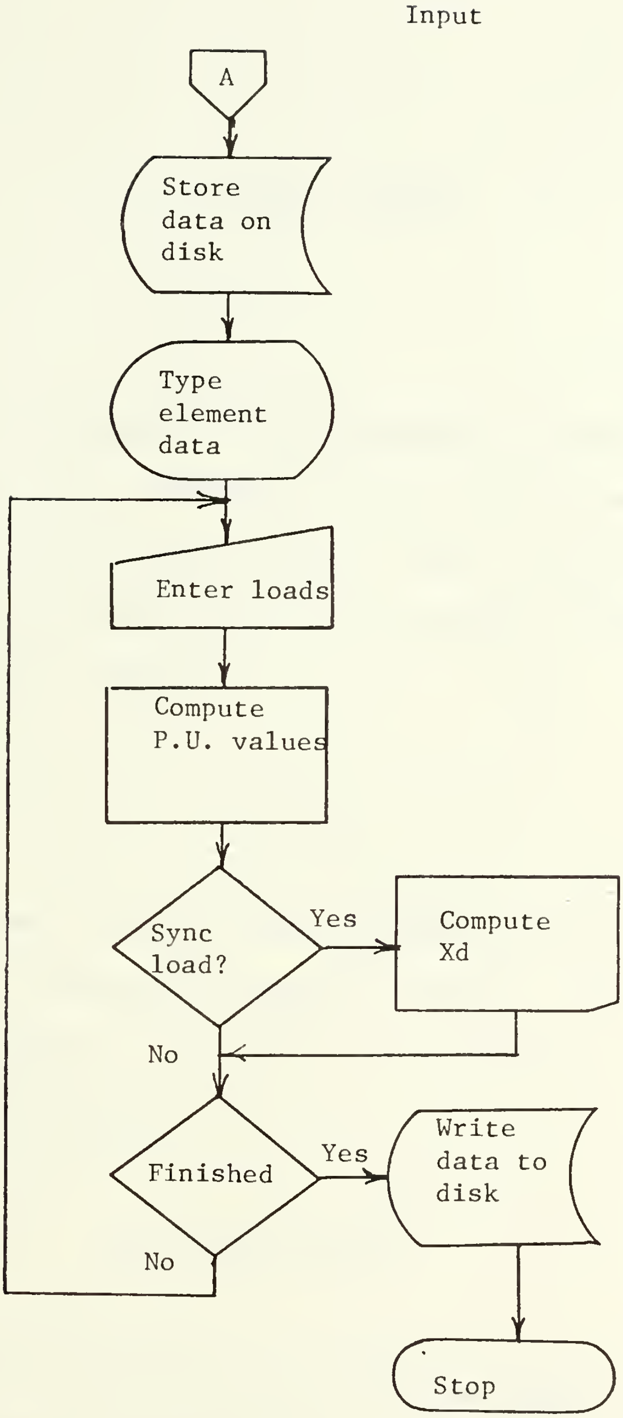


Figure 3.1. Continued.

D
N

4. LOAD-FLOW ANALYSIS

General

In order to establish overload conditions on each element, the load currents must be determined. This is done by using a load flow analysis. The Newton-Raphson Method using Y-bus [12] was chosen because it required the least amount of iterations to obtain a solution to the load-flow problem.

Load Flow

In this method Y-bus must first be formed from the primitive-impedance network. When mutual coupling is ignored, Y-bus can be formed easily by algorithm. Any system can be completely represented by a set of equations [8,12].

$$I^{bus} = (Y^{bus})(E^{bus}) \tag{4.1}$$

E^{bus} and I^{bus} are column matrices with one entry for each bus in the system. Y^{bus} is a square matrix of order equal to the number of buses in the system. Therefore, once all bus voltages are known and Y-bus is formed, matrix multiplication gives the desired load currents.

D
N

However, there is no direct approach to obtain the bus voltages. The only approach to use is that of iteration. The Newton-Raphson method accelerates the iteration process so that normally an accurate answer is obtained after three iterations.

The power at bus p is (from 4.1):

$$P_p - jQ_p = E_p^* I_p$$

$$P_p - jQ_p = E_p^* \sum_{q=1}^{n} Y_{pq} E_q \qquad (4.2)$$

$$E_p = e_p + jf_p \quad Y_{pq} = G_{pq} - jB_{pq} \qquad (4.3)$$

So

$$P_p = \sum_{q=1}^{n} \{e_p(e_q G_{pq} + f_q B_{pq}) + (f_q G_{pq} - e_q B_{pq})\} \qquad (4.4)$$

$$Q_p = \sum_{q=1}^{n} \{f_p(e_q G_{pq} + f_q B_{pq}) - e_p(f_q G_{pq} - e_q B_{pq})\} \qquad (4.5)$$

The Newton-Raphson method uses the Jacobian matrix of partial-differential equations as follows:

$$\begin{bmatrix} \Delta P_1 \\ \Delta P_{n-1} \\ \hline \Delta Q_1 \\ \Delta Q_{n-1} \end{bmatrix} = \left[\begin{array}{ccc|ccc} \frac{\partial P_1}{\partial e_1} & - - - - & \frac{\partial P_1}{\partial e_{n-1}} & \frac{\partial P_1}{\partial f_1} & - - - - & \frac{\partial P_1}{\partial f_{n-1}} \\ \frac{\partial P_{n-1}}{\partial e_1} & - - - & \frac{\partial P_{n-1}}{\partial e_{n-1}} & \frac{\partial P_{n-1}}{\partial f_1} & - - - & \frac{\partial P_{n-1}}{\partial f_{n-1}} \\ \hline \frac{\partial Q_1}{\partial e_1} & - - - & \frac{\partial Q_1}{\partial e_{n-1}} & \frac{\partial Q_1}{\partial f_1} & - - - & \frac{\partial Q_1}{\partial f_{n-1}} \\ \frac{\partial Q_{n-1}}{\partial e_{n-1}} & - - - & \frac{\partial Q_{n-1}}{\partial e_{n-1}} & \frac{\partial Q_{n-1}}{\partial f_1} & - - - & \frac{\partial Q_{n-1}}{\partial f_{n-1}} \end{array}\right] \begin{bmatrix} \Delta e_1 \\ \Delta e_{n-1} \\ \hline \Delta f_1 \\ \Delta f_{n-1} \end{bmatrix}$$

$$\begin{bmatrix} \Delta P \\ \hline \Delta Q \end{bmatrix} = \left[\begin{array}{c|c} J_1 & J_2 \\ \hline J_3 & J_4 \end{array}\right] \begin{bmatrix} \Delta_e \\ \hline \Delta_f \end{bmatrix} \qquad (4.6)$$

The elements of the Jacobian matrix are found from the following equations:

$$J_1: \frac{\partial P_p}{\partial e_q} = e_p G_{pq} - f_p B_{pq} \quad q \neq p$$

$$\frac{\partial P_p}{\partial e_p} = 2e_p G_{pp} + f_p B_{pp} - f_p B_{pp} + \sum_{\substack{q=1 \\ q=p}}^{n} (e_q G_{pq} + f_q B_{pq}) \qquad (4.7)$$

$$J_2:\ \frac{\partial P_p}{\partial f_q} = e_p B_{pq} + f_p G_{pq} \quad q \neq p$$

$$\frac{\partial P_p}{\partial f_p} = e_p B_{pp} + Z_{fp} G_{pp} - e_p B_{pp} + \sum_{\substack{q=1 \\ q \neq p}}^{n} (f_q G_{pq} - e_q B_{pq}) \tag{4.8}$$

$$J_3:\ \frac{\partial Q_p}{\partial e_q} = e_p B_{pq} + f_p G_{pq} \quad q \neq p$$

$$\frac{\partial Q_p}{\partial e_p} = f_p G_{pp} - f_p G_{pp} + Z_{ep} B_{pp} - \sum_{\substack{q=1 \\ q \neq p}}^{n} (f_q G_{pq} - e_q B_{pq}) \tag{4.9}$$

$$J_4:\ \frac{\partial Q_p}{\partial f_q} = e_p G_{pq} + f_p B_{pq} \quad q \neq p$$

$$\frac{\partial Q_p}{\partial f_p} = e_p G_{pp} + Z_{fp} B_{pp} - e_p G_{pp} + \sum_{\substack{q=1 \\ p \neq q}}^{n} (e_q G_{pq} + f_q B_{pq}) \tag{4.10}$$

Bus #1 or the swing-bus voltage is known. The swing bus is the bus that will provide whatever power the system requires. The program begins by forming Y-bus by using the primitive-impedance matrix generated by INPUT.F4 of Chapter 3. Since mutual coupling is ignored, the y-admittance matrix can be found by inverting each member of the z-impedance matrix found in disk file TEST.DAT. Y^{bus} matrix is formed using the algorithm below.

Diagonal elements are formed as the sum of all admittances connected to that bus or node. Off-diagonal elements are the negative of the admittance connected between the buses or nodes.

If more than 10 iterations are performed, the program stops and informs the user that either an error has been made in design or data entry. Upon successful completion of the program, a load-flow analysis may be obtained by printing file DISK.DAT. The program puts the load currents in the proper format and file for later use in the coordination program. A flow chart follows in Figure 4.1, and a complete listing is given in Appendix B.

Summary

This load-flow analysis obtained can be very useful in analyzing the distribution-system design. It can be used to point out locations for power factor correction as well as potential problems due to undersized wiring.

The load currents found here are stored on disk to be used in the coordination program. The next step before beginning the

coordination program is to solve for all fault currents. Z-bus must be formed to compute these currents. The methods used to do this follows in the next chapter.

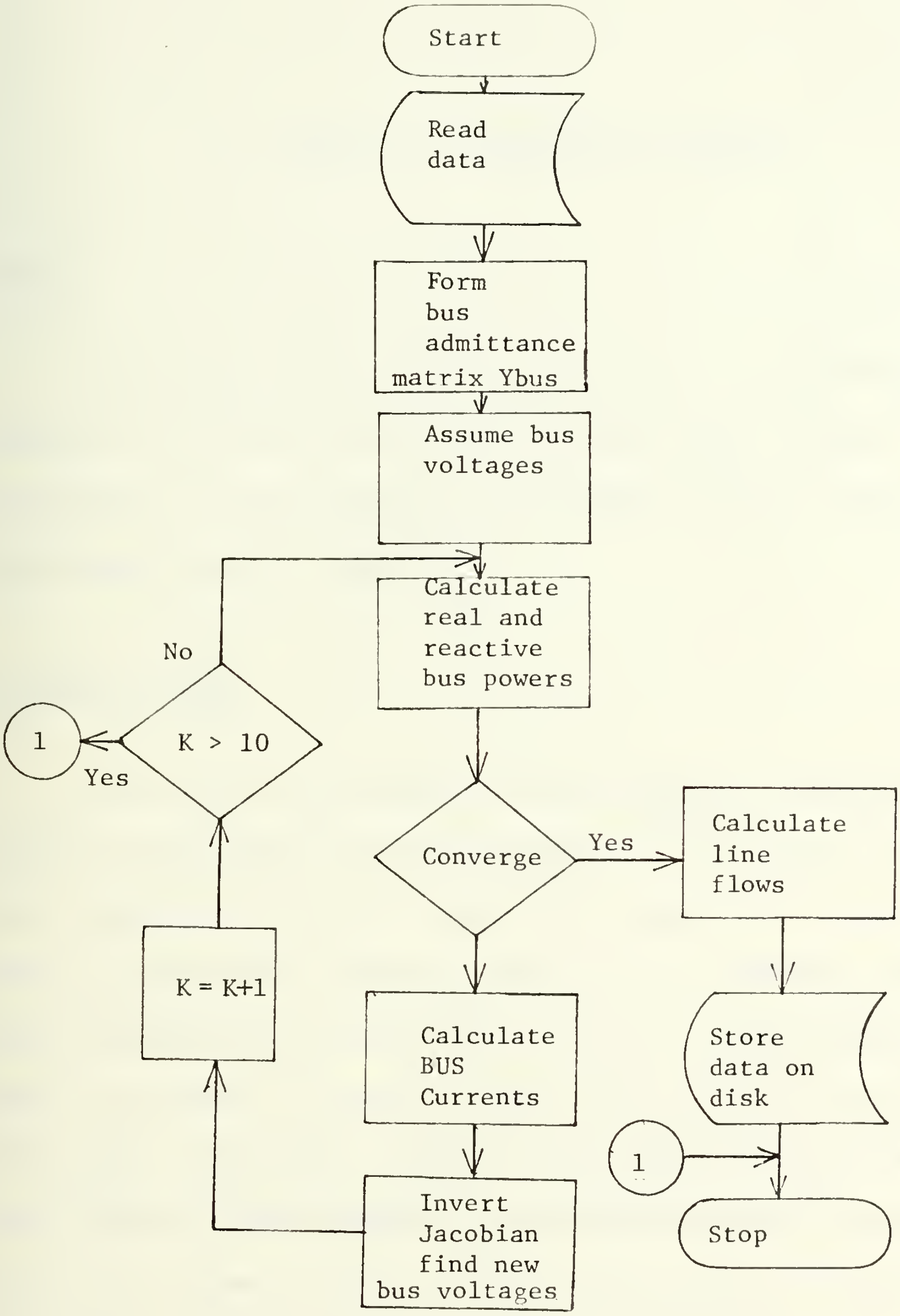


Figure 4.1. Newton-Raphson load-flow chart.

5. FORMATION OF IMPEDANCE-BUS NETWORK

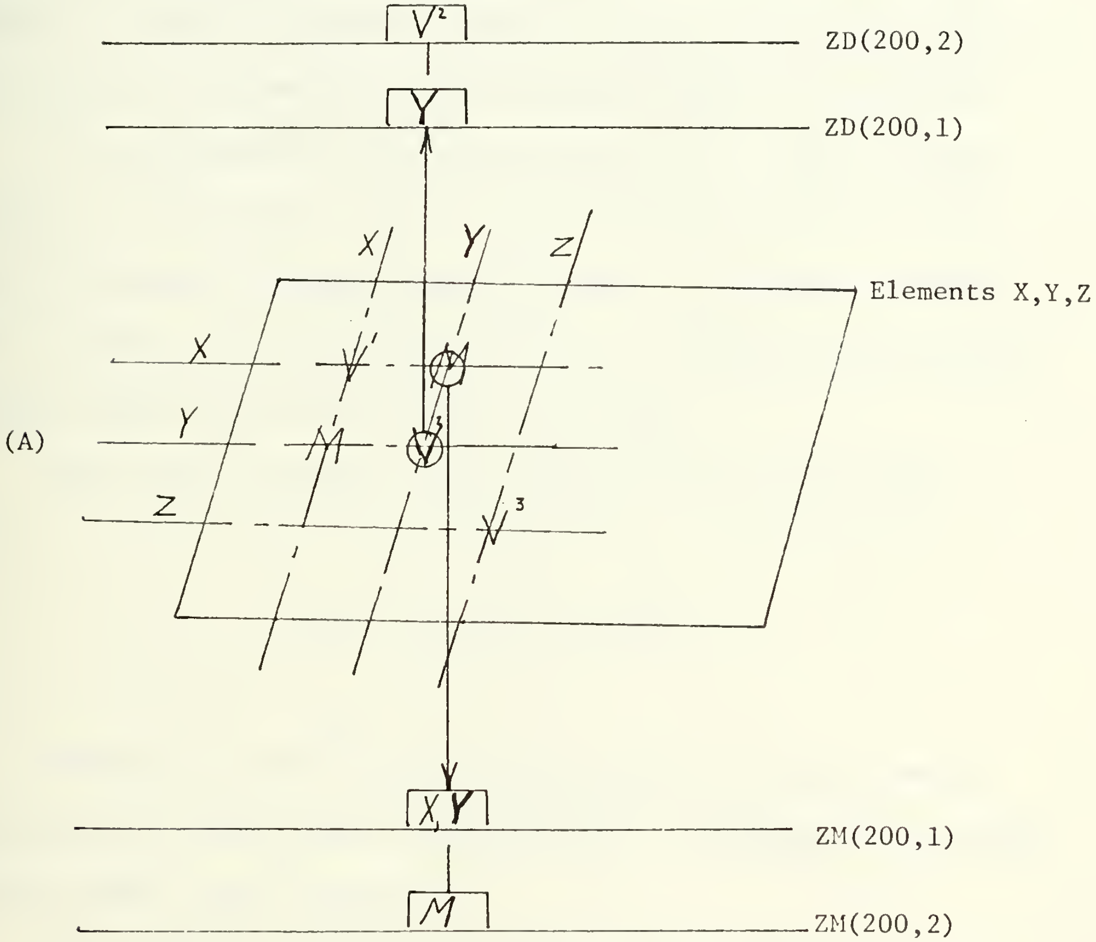
General

The formation of the impedance bus network (Z-Bus^{012}) by algorithm is the first step toward solving for fault currents. In this chapter this algorithm will be discussed as well as other problems encountered in matrix compression.

Formation of Z-Bus

Once the one line diagram has been completed, the primitive impedance matrix that was found in INPUT.F4 is a partially filled matrix. However, it is not in the proper format for the formation of Z-bus as it contains no coupling. The primitive impedance matrix must change from one of order equal to the number of buses to one of order equal to the number of elements. This matrix becomes a very large sparse matrix. The memory required to store this matrix becomes too large to fit in a small computer. A method was derived to only store the non zero values and their location in the matrix. Since the matrix is triangular only the top half is stored (Figure 5.1).

Diagonal elements found in INPUT.F4 are stored in "ZD." ZD is a three-dimensional matrix, ZD(200,2) where (X,1) is the element number and (X,2) is the diagonal value.



Number locations matrix (A) = (# elements)^2 X 2
Number of locations of alternate method = 8 X (# elements)

Figure 5.1. Transformation of square sparse primitive impedance.

The off diagonal mutual coupling Zm is a three dimensional matrix also where (X,1) is the location value A+jB. Element A is coupled to element B. Location (X,2) is the coupling value. Fortunately, partitioning makes it necessary to assemble a square matrix for inversion of only the coupled elements. Since coupling only exists in zero sequence, this applies only to the zero-sequence Z-bus matrix.

In the formation of the Z-bus matrix, bus 1 is the reference bus for all calculations. Z-bus is formed by starting with element one and proceeding with each element in turn. An element is either a link or branch. A link means that both buses have been used earlier; a branch means they have not been used earlier.

Like Y-bus in Chapter 4 any system can be represented by

$$\mathrm{Ebus}^{012} = \mathrm{Zbus}^{012}\ \mathrm{Ibus}^{012}$$

Formation of Zbus^{012} can be formed either by the incidence, network-matrices method [12] or by algorithm. Formation by algorithm is much simpler and is used here.

After forming the primitive impedance network, it is a straightforward application of the equations of Table 5.1.

Table 5.1. Equations for the formation of ZBUS. [11]

Add p-q	Mutual Coupling	No Mutual Coupling
Branch	$Z_{qi} = Z_{pi}+\bar{y}_{pq,rs}(\bar{Z}_{ri}-\bar{Z}_{si})$	$Z_{qi} = Z_{pi}$
	i = 1, 2, . . . , m i ≠ q	
	$Z_{qq} = Z_{pq}+\dfrac{1+\bar{y}_{pq,rs}(\bar{Z}_{pq}-\bar{Z}_{rs})}{y_{pq,rs}}$	$Z_{qq} = Z_{pq}+z_{pq,pq}$
Link	$Z_{li} = Z_{pi}-Z_{qi}+\dfrac{\bar{y}_{pq,rs}(\bar{Z}_{ri}-\bar{Z}_{si})}{y_{pq,pq}}$	$Z_{li} = Z_{pi}-Z_{qi}$
	i = 1, Z, . . . , m i ≠ l	
	$Z_{ll} = Z_{pl}-Z_{ql}+\dfrac{1+\bar{y}_{pq,rs}(\bar{Z}_{rl}-\bar{Z}_{sl})}{y_{pq,pq}}$	$Z_{ll} = Z_{pl}-Z_{ql}$ $+ z_{pq,pq}$
Elimination of l^th Node		
	$Z_{ij}(\text{modified}) = Z_{ij}(\text{before elimination}) - \dfrac{Z_{il}Z_{lj}}{Z_{ll}}$	
	i,j = 1, 2, 3, . . . , m	

For example, a primitive impedance matrix looks like this for five elements:

$$z^o = \begin{array}{c|ccccc|} & 1(1-2) & 2(2-3) & 3(3-4) & 4(1-4) & 5(2-4) \\ 1 & .1+j.5 & j5 & & & \\ 2 & j5 & .1+j.5 & & & \\ 3 & & & .1+j.5 & j10 & j4 \\ 4 & & & j10 & .1+j.5 & j8 \\ 5 & & & j4 & j8 & .1+j.5 \end{array} \quad y^o = [z^o]^{-1}$$

where the off diagonal values represent mutual coupling between elements. Using this primitive matrix and the equations in Table 5.1 yields Zbus.

ZBUS.F4 is one of the longest programs in the series. The program asks for the zero-sequence impedance for all loads in the system and the mutual-coupling impedances. If the zero-sequence impedance of a load is not known, the program assigns it the same value as the positive sequence impedance.

All loads that are added are links, and only loads that are added are those of synchronous motors. The subtransient reactance is used instead of its actual impedance since the Z-bus is going to be used for fault calculations. Any link from a bus to the reference bus will cause a fault current. Prefault voltages and currents will be ignored as will all contributions from loads other than synchronous motors. A flow chart follows (Figure 5.2) and a complete program listing in Appendix B.

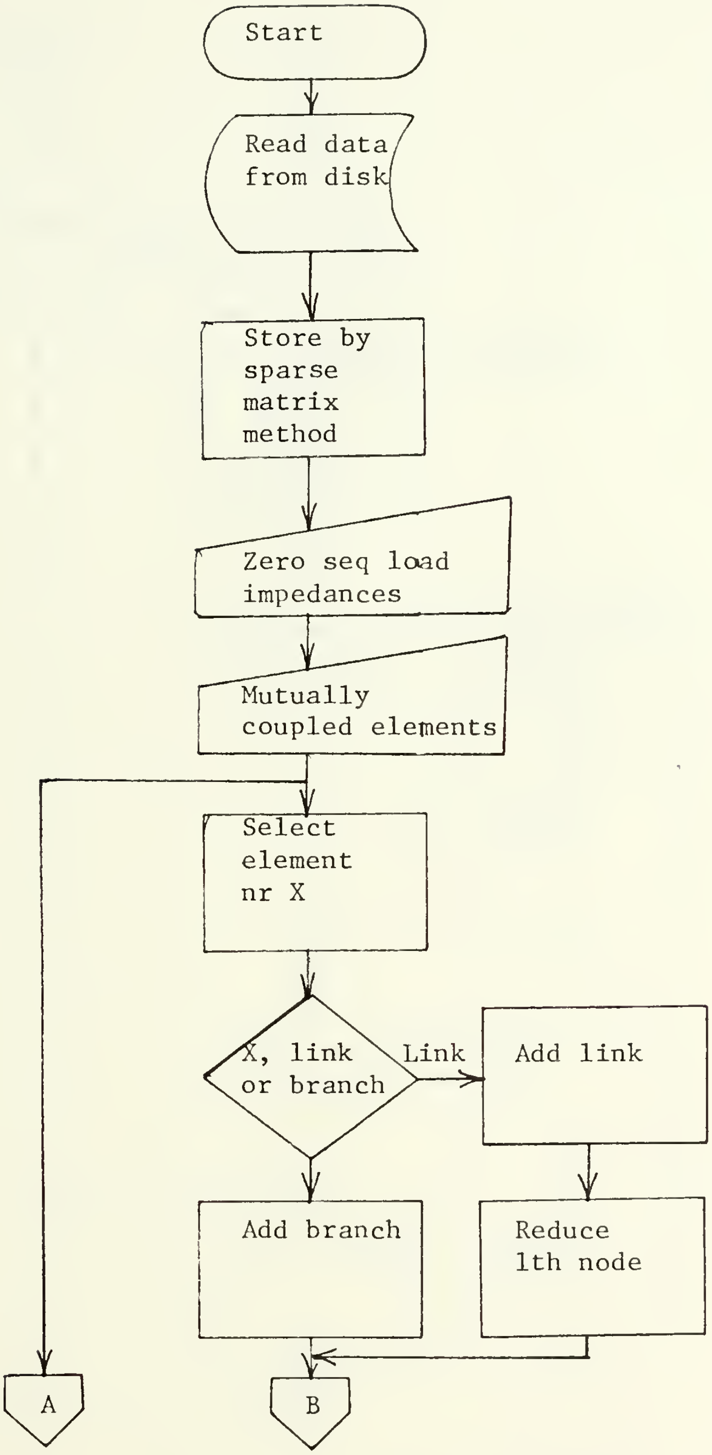


Figure 5.2. Zbus^{012} formation flow chart.

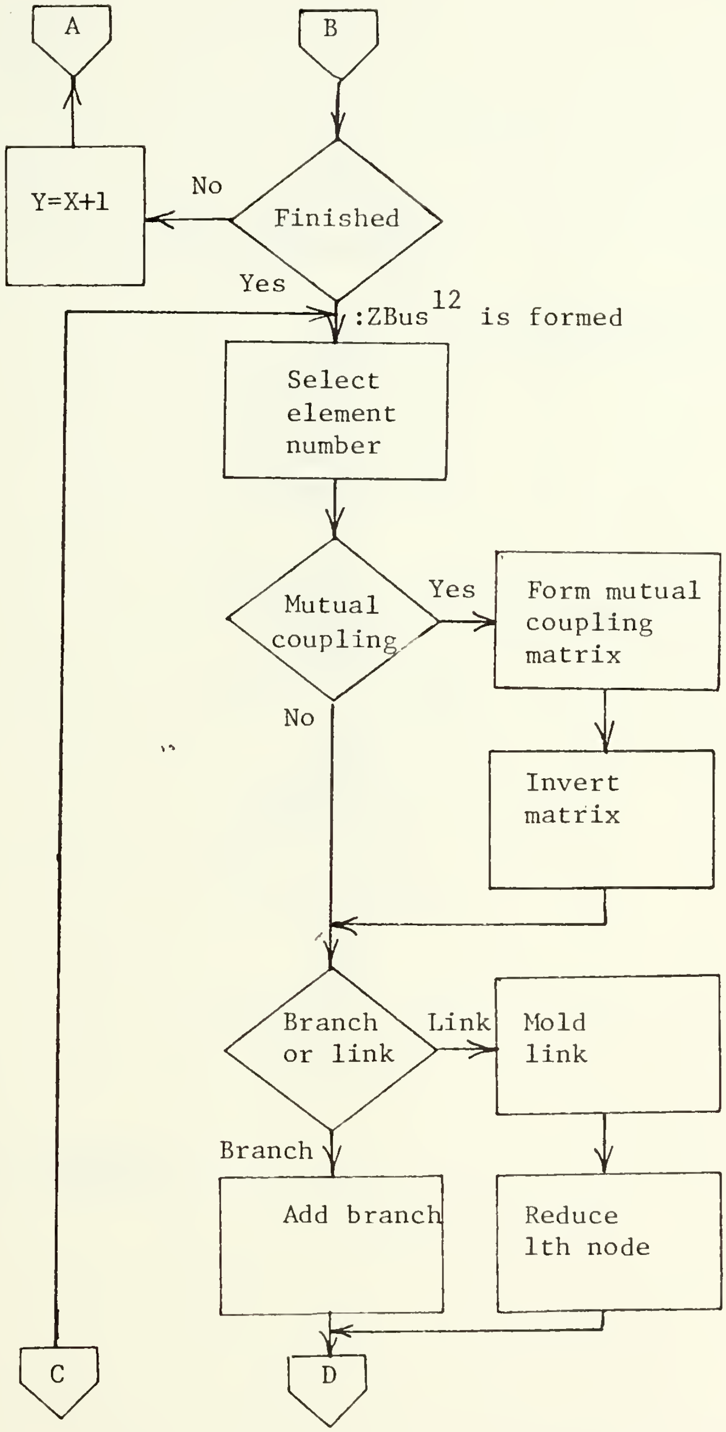


Figure 5.2. Continued.

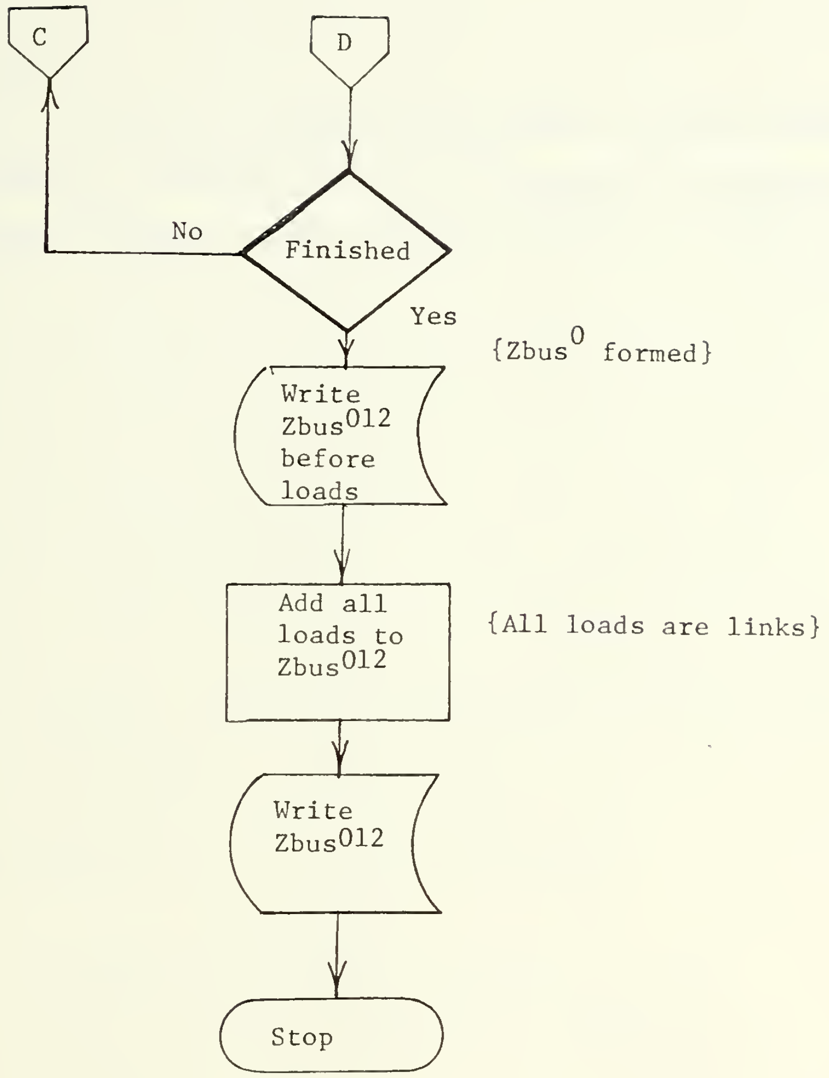


Figure 5.2. Continued.

Summary

$Zbus^{012}$ is formed by algorithm allowing for mutual coupling between elements. The next step in the coordination procedure is to solve for fault currents. This procedure follows in the next chapter.

6. FAULT CALCULATIONS

General

The final program required before executing the coordination program is FALT.F4. This program computes all the possible line-to-ground and three-phase-fault currents and prepares all other load data for entry into the coordination program. The procedure to accomplish this will be discussed in this chapter.

Fault Currents. The equations describing the fault currents need to be derived [12].

Three-Phase Fault. For a fault at bus P:

$$I_{P(F)}^{012} = Y_F^{012}(U+Z_{pp}^{012}Y_F^{012})^{-1}E_{P(0)}^{012} \tag{6.1}$$

where U is unity matrix, $E_p(0)$ is prefault voltage, and Y_F is the fault admittance matrix for a three-phase fault.

$$Y_F^{012} = y_F \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6.2}$$

Substituting (6.2) into (6.1)

$$I_{P(F)}^{012} = y_F \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1+Z_{pp}^{o} & & \\ & 1+Z_{pp}^{1} y_f & \\ & & 1+Z_{pp}^{2} y_f \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ \sqrt{3} \\ 0 \end{bmatrix}$$

$$I_{P(F)}^{012} = y_F \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{1+Z_{pp}^{o}} & & \\ & \frac{1}{1+Z_{pp}^{1} y_F} & \\ & & \frac{1}{1+Z_{pp}^{2} y_F} \end{bmatrix} \begin{bmatrix} 0 \\ \sqrt{3} \\ 0 \end{bmatrix}$$

$$I_{P(F)}^{012} = y_F \begin{bmatrix} 0 & 0 & 0 \\ 0 & \frac{1}{1+Z_{pp}^{1} y_F} & 0 \\ 0 & 0 & \frac{1}{1+Z_{pp}^{2} y_F} \end{bmatrix} \begin{bmatrix} 0 \\ \sqrt{3} \\ 0 \end{bmatrix} \qquad (6.3)$$

Since $Zpp^1 = Zpp^2$ and $z_f = 1/y_f$

$$I_{P(F)}^{1} = \frac{\sqrt{3}}{z_F + Z_{PP}} \quad I_P^0 = 0 \quad I_P^2 = 0 \qquad (6.4)$$

Using the same method the following equations are found [12].

$$E^{1}_{P(f)} = \frac{\sqrt{3}\, z_F E_{Po}}{z_F + Z_{PP}^{1}} \qquad (6.5)$$

and

$$E_{i(F)}^{1} = \sqrt{3}\,(E_{i(0)} - \frac{Z_{ip}^{(1)} E_p(0)}{z_f + Z_{pp}^{(1)}}$$

$$i \neq P \quad E_i^{0,2} = 0 \quad E_{p(f)}^{0,2} = 0 \qquad (6.6)$$

These equations are identical to those found for a three-phase grounded fault [12].

The line-to-ground fault admittance matrix is given by Y_F^{012} below. Substituting Y_F^{012} into (6.1) the equations below result [12].

$$Y_F^{012} = \frac{y_F}{3} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$I_P^{012} = \frac{\sqrt{3}\, E_{p(0)}}{Z_{pp}^{0} + 2Z_{pp}^{(1)} + 3z_F} \qquad (6.7)$$

$$E_p^{012} = \frac{\sqrt{3}\, E_p^{0}}{Z_{pp}^{(0)}+2Z_{pp}^{(1)}+3z_F} \begin{bmatrix} -Z_{pp}^{(0)} \\ Z_{pp}^{(0)}+Z_{pp}^{(1)}+3z_F \\ -Z_{pp}^{(1)} \end{bmatrix} \qquad (6.8)$$

$$E_{i(F)}^{012} = \underset{i \neq p}{E_{i(0)}} \begin{bmatrix} 0 \\ \sqrt{3} \\ 0 \end{bmatrix} - \frac{\sqrt{3}\, E_{p(0)}}{Z_{pp}^{(0)}+2Z_{pp}^{(1)}+3z_F} \begin{bmatrix} Z_{ip}^{(0)} \\ Z_{ip}^{(1)} \\ Z_{ip}^{(1)} \end{bmatrix} \qquad (6.9)$$

The value of z_F is entered from the keyboard when asked by the program. All of the equations are available now to solve for all fault currents. All post-fault bus voltages are solved. Using these bus voltages each element fault current is determined. These currents are stored onto disk so that they may be used in both the coordination and graphing programs.

Summary

The programs thus far have solved for all load and fault currents but no coordination has been discussed. Now the coordination problem is ready to be addressed. In the chapter that follows, the protective devices for the distribution system will be coordinated.

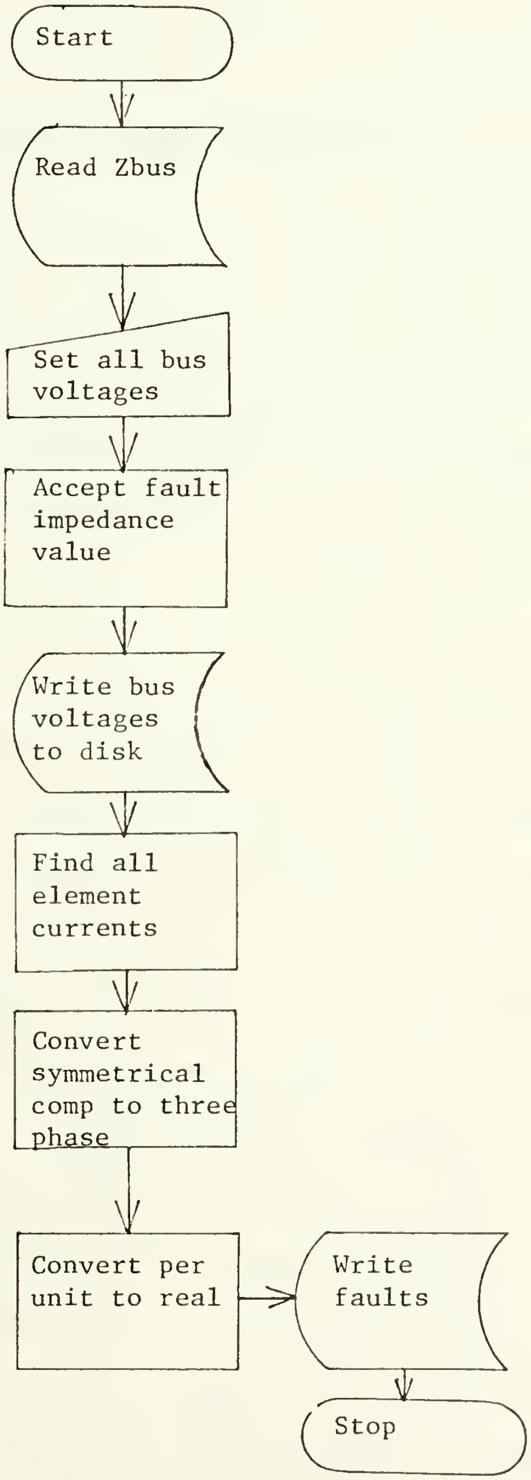

Start
Read Zbus
Set all bus voltages
Accept fault impedance value
Write bus voltages to disk
Find all element currents
Convert symmetrical comp to three phase
Convert per unit to real
Write faults
Stop

Figure 6.1. Flow chart for FALT.F4.

7. PROTECTIVE-DEVICE COORDINATION

General

In this chapter, the protective-device-coordination problem will be discussed. A computational method is put forward that allows a digital computer to solve the problem.

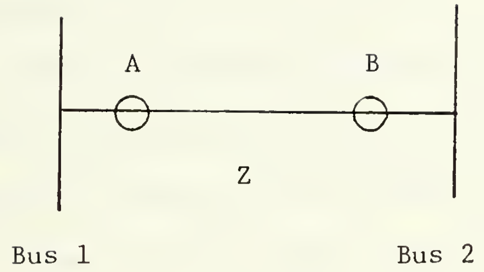
Device Coordination

One of the best methods to approach coordination is a step by step device selection and setting working from the load toward the source. A load is the best place to start because there is not much flexibility for device setting at a load. The load flow analysis of Chapter 4 provides the overload values for the various loads. This enables a definite overload setting for each device protecting a load. Since overload currents are a primary concern at these buses, all fault settings can be placed at six times the full load currents. This should allow for normal starting currents. Care must be taken so that an instantaneous tripping element will not open under normal starting conditions. From the loads it is a straightforward process to place a line side device curve next to its load side curve to insure that proper coordination is achieved. By taking this approach, each device depends only on one other device.

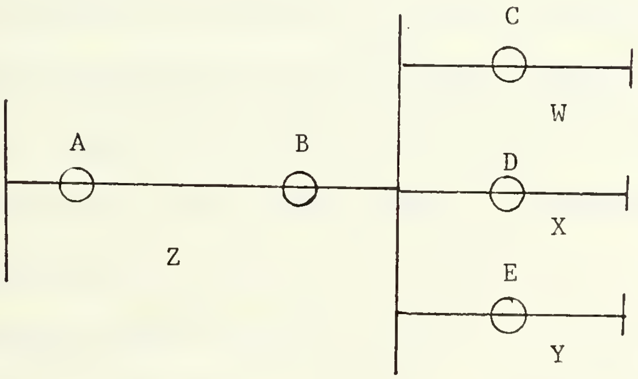
This program is written with the option of entering two devices in each element--one located at each bus. From this approach, obviously two possibilities can occur. Either the device is on the line side of the element or on the load side of the element (Figure 7.1). If it is the line-side device, its setting depends only on the one load-side device. It must be coordinated for the fault current at the load bus and not the line bus [1]. Devices must be coordinated for a minimum possible fault current which will be discussed later.

If the device is on the load side and it does not protect a load, then it must protect a following bus. This device must then be coordinated with all line-side devices connected to the bus it protects. This load-side device must be coordinated for fault currents at the protected bus.

Devices should be set to (1) open under overload conditions, (2) open under minimum fault conditions with proper coordination, and (3) open under maximum fault conditions with proper coordination. Often, however, all three conditions cannot be met simultaneously unless the proper devices are chosen. Criterion (1) is not as important as Criterion (2) or (3) when in the distribution system not at a load bus since an overload condition below fault conditions should not do damage to the wiring. For this reason wire sizes must be checked to insure that upon final coordination they can withstand the overload and fault conditions that can occur [5,13]. At this point it may be necessary to begin the entire coordination procedure again if an element cannot withstand the short-circuit conditions. Allowable short-circuit currents can be checked by using the final coordination graph and the fault currents obtained in FALT.F4.



(a) Coordinating line side device of an element.



(b) Coordinating load side device of an element feeding a bus with load elements connected W, Z. Y.

Figure 7.1. Two possibilities for coordination element Z. In Figure (7.1a) coordination device A according to curve device B up to fault bus 2. In Figure (7.1b) coordination device B depending on load flow current element Z. Coordinated with device curves C, D, E, up to fault at Bus 2.

This coordination routine utilizes fault currents and load currents in achieving protective-device coordination. The load currents determine the overload or pickup setting in each device. The coordination problem is then in two parts.

Devices must be coordinated for overload conditions. This provides the most important coordination step for fuses in that they contain no variable fault-protection settings as do relays or molded-case circuit breakers. Also, another reason is that based upon only fault currents and load-side device settings, it is impossible to satisfactorily achieve coordination.

This inability to achieve coordination enters when setting a device that is to protect a bus with other elements attached to that bus as loads. Just laying a line curve next to a load curve with the highest time value at any current value will provide an erroneous answer. This device must be set by determining its overload current value first since this current is the sum of all currents in the load elements that are attached to that bus. Once this point has been established, each load device curve must be polled in turn to insure that the device setting responds to a fault in a load element slower than the load element line side device. The device needs to be coordinated up to the maximum fault current that the device will see. It is possible for this device to see a smaller fault current than that in a load element.

Coordinating a device located at a line-side bus is much simpler since the pickup or overload setting is the same as the load-side device. The line-side device must be coordinated up to the maximum

fault current that will occur at the bus where its load-side device is located. A higher fault current will occur the closer one gets to the line-side device; however, this higher current value will only cause the line-side device protecting the element to respond quicker. The method of coordination will be to first find a load point based on a load flow, and second coordinate the device at any current value up to some maximum fault current such that the line-side device has a longer response time than its load-side device.

Since the program coordinates three different types of devices, nine different possibilities can occur. Each device may be coordinated with one of its own type or one of the other two.

The program structure lends itself well to the use of subroutines. Two of these are SELECT and SETDEV. SELECT chooses a device and sets its own load point, and SETDEV coordinates it.

Fuse Selection and Coordination

Due to the large number of fuse sizes available, look-up tables are not as efficient as using polynomial functions to select devices. Once a load current is known, this value is substituted into one function providing the proper device number. This device number is next substituted into two other functions providing the OFSET and OFSET 1 values referred to in Chapter 2. Since there are two types of fuses in the program, current-limiting and solid-material boric-acid power fuses, there are six functions used in selecting fuses. These all have the following form:

$$Y = \sum_{j=0}^{n} a_j X^j \qquad (7.1)$$

X will be the logarithm of overload current or device number. Table 7.1 lists the various coefficients a_j.

After the fuse is selected, it must be coordinated. The program uses the values of OFSET and OFSET 1 with the applicable fuse curve to solve for a time value for a set-current value. The program searches for the load-side device to find the highest time value to that current value. If the time value is larger for the size, it is properly coordinated. If not, it increases the fuse size by one, solves for OFSET and OFSET 1, and tries again continuing until it has found the proper fuse size.

Relay Selection and Coordination

Relays provide the most difficult coordination problem. First a turns ratio for the current transformer (CT) must be chosen. The computer selects the proper tap setting to prevent saturation. It does this by always selecting the highest turns possible while selecting the tap setting on the relay as well [12]. The computation uses a 600/5 CT [13] with tap settings and CO relays [14] with tap settings. This combination of tap setting for the CT and the relay combined with the load current establish the pickup point:

$$\text{Pickup} = \text{Log}\ [(\text{Turns Ratio CT} \times \text{Turns Ratio Relay})] \qquad (7.2)$$

Table 7.1. Fuse-selection coefficients.

Current to Device		Offset	Offset 1
Current Limiting Fuses			
a_0	- 11.15221	.9532	1.1619
a_1	23.865	- .046433	- .110148
a_2	- 16.487	.117885	.10626
a_3	2.9288	- .029352	- .02249
a_4	3.19672	.00337	.002452
a_5	- 1.48877	- .0001847	- .0001365
a_6	.1776431	.0000039	.00000304
Solid Material Boric Acid			
a_0	88.1643	1.34359	1.41734
a_1	-215.2235	.151393	.1585889
a_2	188.225	- .0239735	- .035007
a_3	- 69.13403	.00046627	.007180
a_4	7.4695	.0004321	.0008126
a_5	1.65757	.0000182	- .0000377
a_6	- .347786	.0000078	0

Since relay curves are in units of multiples of pickup this pickup value will be used often. Only the relay pickup has been set, its time dial must be coordinated. Table 7.2 lists the relay-curve coefficients.

Due to the inverse time shape of a relay curve, many points on the curve must be checked with its load-side device to insure that proper coordination exists. The manner in which the family of relay curves are represented simplifies the problem somewhat. A relay operating time of 0.4 seconds is assumed throughout. At each current value, the load-side-device operating time is summed with the relay operating time. If this time is greater than the line-device time, the difference is the relay time-delay setting. Thus, when all currents up to the fault currents are checked, all points on the line-device curve are at least 0.4-seconds higher than all points on its load-side device.

Molded Case Circuit Breaker

The molded-case circuit breaker (MCCB) with its different characteristics provides a different set of problems. The MCCB curve cannot be represented by one equation. This device has one set of curves for the thermal element plus another for its magnetic element. The MCCB used in the program is a General Electric type, K-1200. Although there is some difference in the curve shape for the current ratings, 300-1200 A, they are similar enough to be represented as one type. Each MCCB has one curve corresponding to minimum total clearing time and a maximum total clearing time.

Table 7.2. Relay-curve coefficients.

	Relay Number	a_o	a_1	a_2	a_3	a_4
CO-2	1	- .4401731	-3.207952	3.098054	-1.132646	0.0
CO-5	2	1.415901	-4.556591	7.501177	-6.484299	2.233915
CO-6	3	.83421	-4.284002	7.005048	-5.298087	1.505842
CO-7	4	1.0438914	-4.308733	6.998595	-5.908597	1.894855
CO-8	5	1.7617574	-4.526460	3.226049	- .4676728	- .2548555
CO-9	6	1.476228	-2.512627	-1.278763	3.363051	-1.356438
CO-11	7	2.897519	-4.885438	4.078362	-2.517864	.7629706

The magnetic portion of the time-current curve cannot be represented adequately by a polynomial function, but is represented by:

If Current $\geq$ Magnetic Setting

Minimum Time = 0, Maximum Time = .025 seconds.

As will be seen later, .025-seconds response time is quite fast and usually does not present a problem during later coordination.

The thermal elements of molded-case circuit breakers are susceptible to temperature change so the operating temperature must be entered. The temperature merely shifts the curve to the left or right so it does not affect the curve shape. This offset is added or subtracted from the multiple of current rating when using the thermal curve polynomial functions.

The problem of coordinating the thermal elements is the same as that for fuses. The magnetic element coordination is much simpler in that the magnetic element is set separately. Its load-side device operating time is checked at a current value. If the operating time is greater, the magnetic element setting is increased. The end of the curve for the magnetic element is determined by the maximum fault current at the bus. The total decoupling of the overload from the fault device of a MCCB makes coordination much simpler than the fuse or relay. The more or less proportional-time shape of the curve will ensure that it will be properly coordinated once its proper current rating has been

established. Table 7.3 contains the polynomial coefficients used in the program.

A basic flow chart (Figures 7.2 and 7.3) follows with a complete program listing in Appendix B.

Summary

In this chapter, the coordination problem was solved to allow fuses, molded-case circuit breakers, and relays to function so as to clear a fault in the area of the fault before interrupting the rest of the distribution system. There is enough flexibility in the program to allow device settings to be changed and checked using the program PLOTD.F4. An example problem would be beneficial at this point to show how to run the programs. In the next chapter, a sample distribution system will be coordinated using the coordination programs.

Table 7.3. Molded-case circuit breaker.

$$\log(\text{time}) = \sum_{n=0}^{6} a_n[\log(\text{current}) - (\text{ofset} + .1122)]^n - 2$$

$$\log(\text{time}) = \sum_{j=0}^{6} a_j[\log(\text{current}) - (\text{ofset} + .2315)^j -2$$

$$\text{ofset} = \log(\text{ckt breaker size})$$

Minimum Time:

$a_0 = 5.76908$

$a_1 = -10.88261$

$a_2 = 23.65418$

$a_3 = -34.7922$

$a_4 = 33.7747$

$a_5 = -21.14536$

$a_6 = 6.431778$

Maximum Time:

$a_0 = 5.8596721$

$a_1 = - 4.3943349$

$a_2 = - .50690283$

$a_3 = -10.402739$

$a_4 = -17.0649561$

$a_5 = -13.011543$

$a_6 = - 4.0107239$

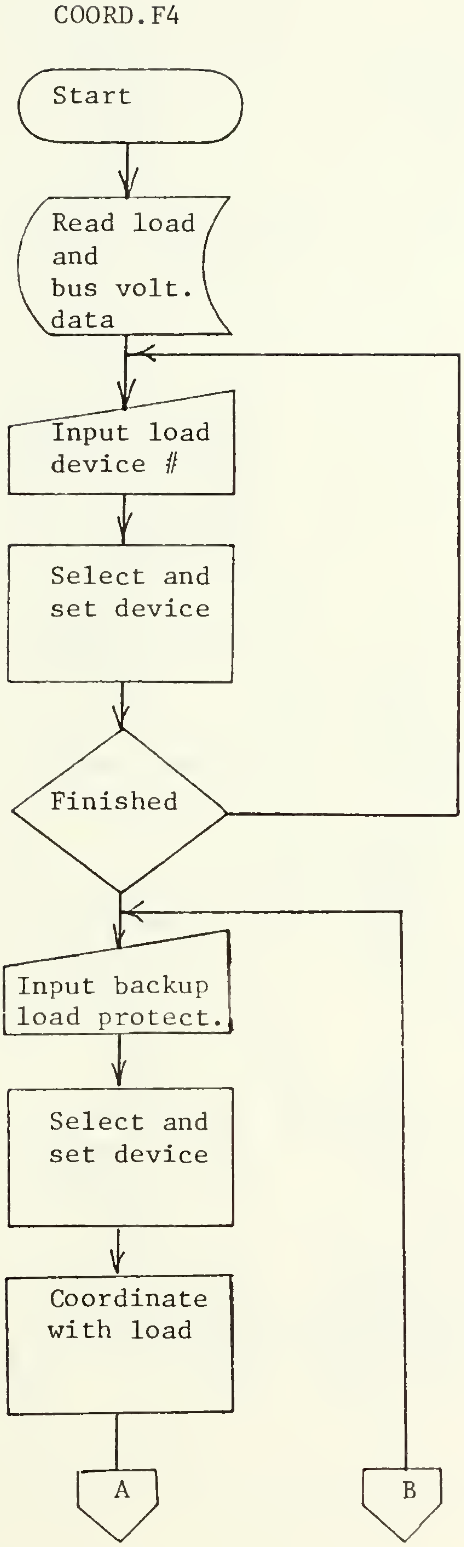


Figure 7.2. Coordination flow chart.

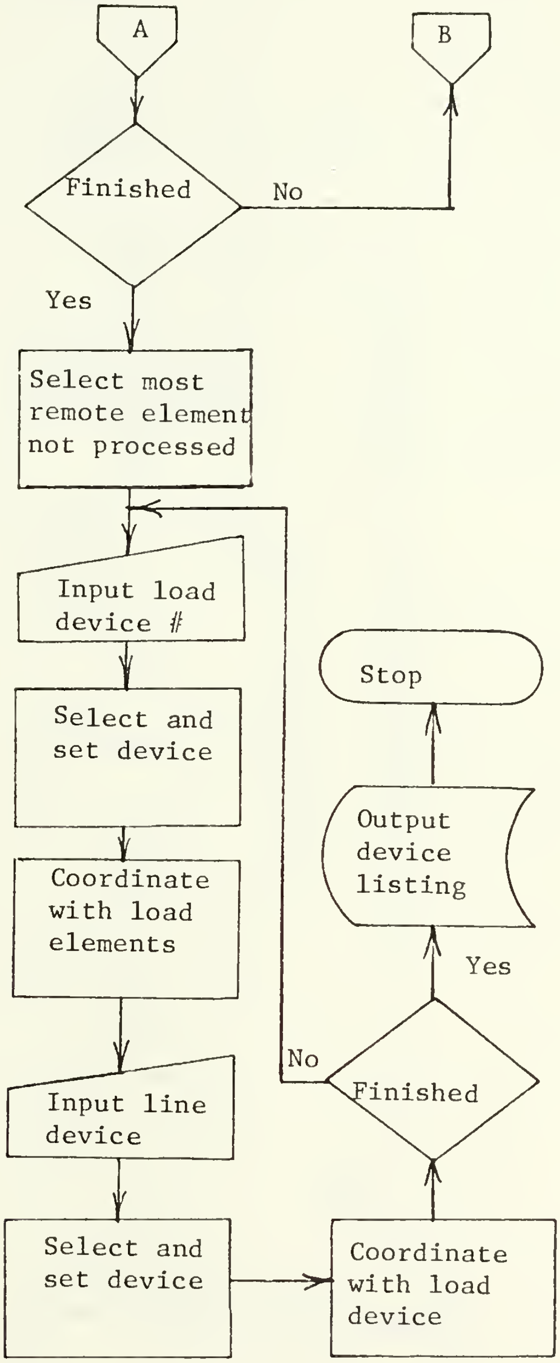


Figure 7.2. Continued.

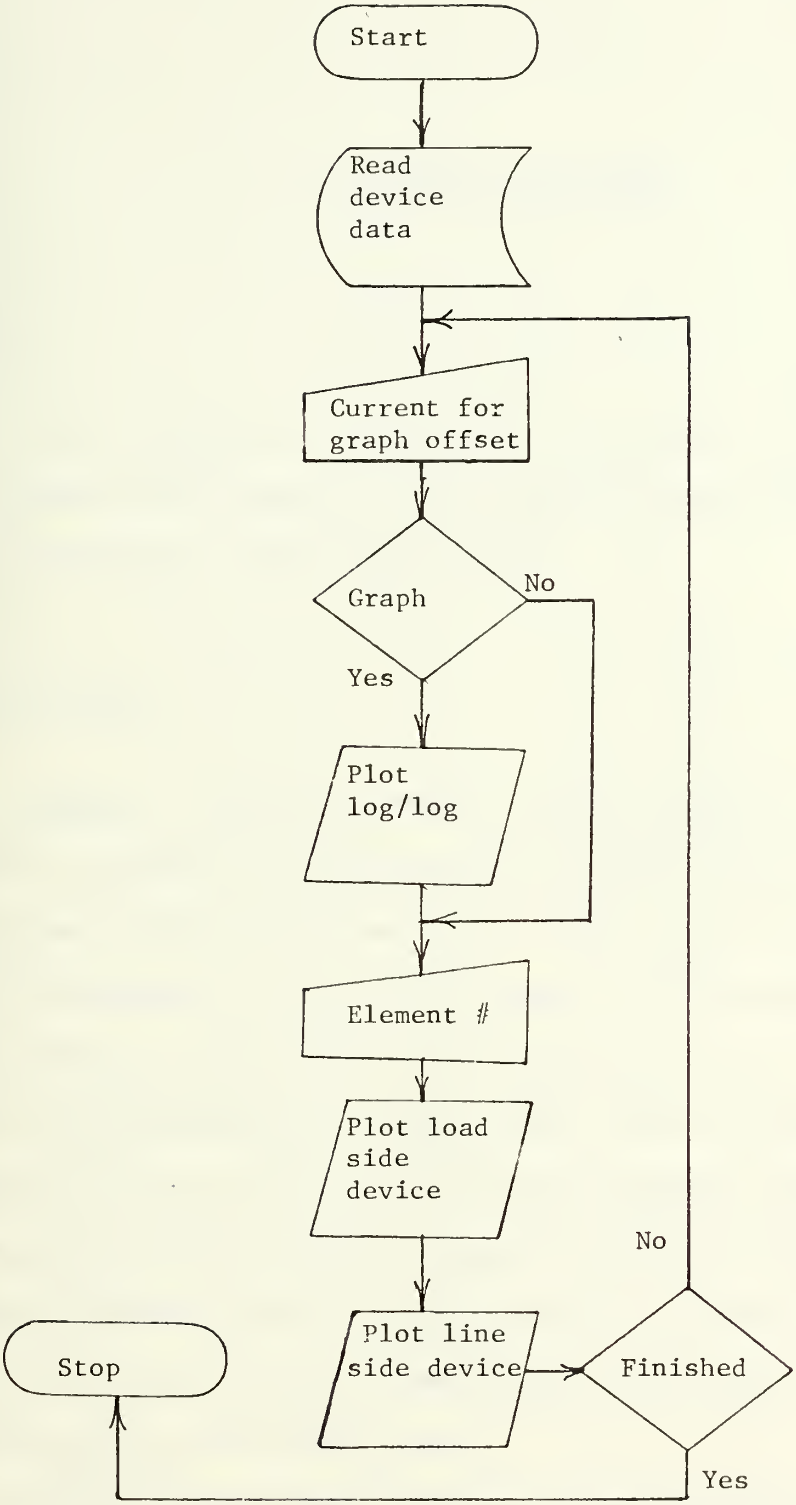


Figure 7.3. Device-graphing flow chart.

8. OPERATING THE PROGRAMS

General

In this chapter the capabilities, limitations, and use of the programs will be described. The example system of Figure 8.1 will be coordinated using these programs.

One-Line Diagram

INPUT.F4. All data is entered via the keyboard in the format detailed in Chapter 3. Data entry is a straightforward process except when entering transformer data. The program is written for a "Y-Y" configuration. If one or both sides are "Δ" configured, then a large number must be entered for the transformer neutral impedance "(ZN)" on the bus connected to the Δ side. After the program is completed, the zero sequence value for the transformer element containing the transformer may be assigned an arbitrary high value if the transformer wiring is such that there can be no ground current through the element. This is done by changing its value in disk file TEST.DAT (see Appendix C for the file format).

When entering the loads the program asks which type of load to be entered. The various loads are described as:

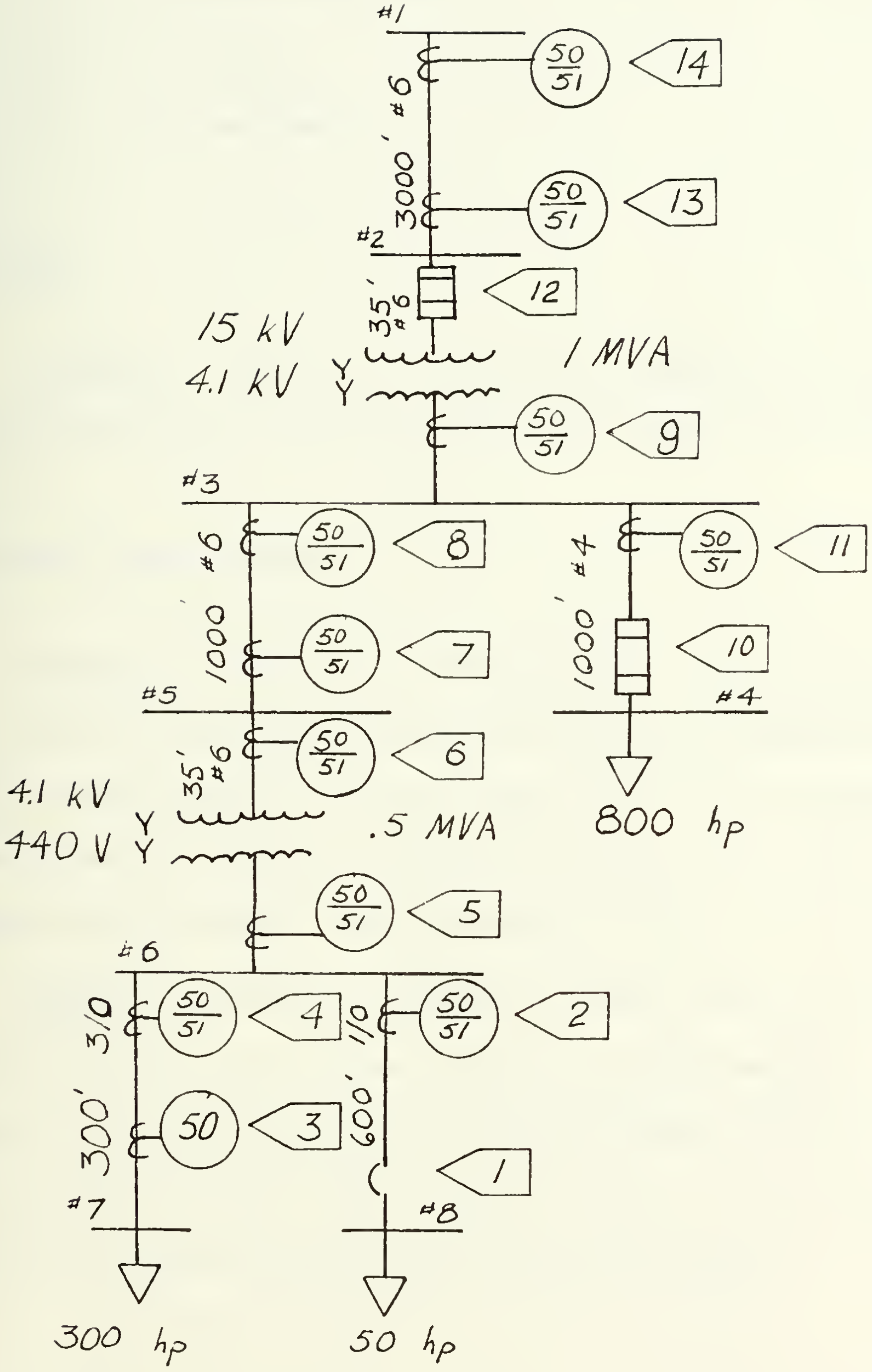


Figure 8.1. One-line diagram of sample system.

HP - Synchronous motor
HP1 - Assynchronous motor
VA - In KVA
CUR - In amperes
S - No more loads

Load Flow

INPUT.F4. There are no entries from keyboard for this program. Printing DSK.DAT will provide a load-flow analysis in per-unit values.

Solution of Fault Currents

ZBUS.F4. In addition to forming Zbus^{012}, this program allows the input for mutual coupling and for zero-sequence impedance for loads. Mutual coupling tends to increase fault currents in the mutually coupled elements and small neutral-to-ground impedances can create higher line-to-ground fault currents than three-phase faults.

FALT.F4. Besides the solution of fault currents described in Chapter 6, FALT.F4 also allows the input of fault impedances. This can provide an interesting study of the system because the user can go back and place different fault impedances into the fault analysis to see if the protective devices will operate as desired.

Device Coordination and Plotting

COORD.F4. The program forms a two-dimensional complex matrix called ADEV that contains all the information required to plot all the devices in the system plus all actual settings that are to be made on the devices. An example row 3 in the matrix is 3,(3,4), (-4,2), (1.26,203), (1.03,1.26), (3,667,0), (.67,0), (20,0) in which:

1. For 3, 3 is the row location of the matrix and corresponds to the element number.

2. For (3,4), it is the first column location and is the bus numbers of the element.

3. For (4,2),-4 is a relay type #4; 2 is a current limiting fuse type #2.

4. For (1.26,203), 1.26 is the logarithm of the pickup point; 203 contains the tap information. Two is the tap of the CT, and 03 is the tap of the relay.

5. For (1.03,1.26), 1.03 is the logarithm of the current for the minimum fuse open; 1.26 is the typical fuse open point.

6. For (3.667,0), 3.667 is the magnetic setting of the relay; 0 since the fuse had no magnetic setting.

7. For (.67,0), .67 is the time-dial setting for the relay. The actual time dial setting must be read from the manufacturer's family of relay curves using the formula below at the maximum fault current.

time-dial-setting found = (found-time curve) -

(time-dial 1/2 setting)

8. For (20,0), 20 is the turns ratio of the current transformer.

Device numbers the programs asks for correspond to the devices below:

1 - Current limiting fuse
2 - Boric fuse high voltage only
3 - Molded-case circuit breaker
4 - Overcurrent relay

In device 4 the program will ask later what particular type of relay to use from Table 8.1.

Upon completion of coordination, a device listing is typed on the keyboard, and a printout of the same is available by typing: PRINT*.LPT.

When performing the plot, having a device listing as well as a fault current listing is helpful.

PLOTD.F4. This program plots the devices that were coordinated above. By using a graph, it is possible to visually check coordination. The program arrangement allows the user to change any device setting or size in ADEV.DAT to tailor the coordination to his particular requirements (see Appendix C for the file format). The program uses the lowest

Table 8.1. Overcurrent-relay types.

Program Type #	Relay Type
1	CO-2
2	CO-3
3	CO-4
4	CO-5
5	CO-6
6	CO-7
7	CO-11

Relay type numbers are listed as negative numbers by the program.

voltage in the system as a base for plotting currents. The user can select any plot offset. This has the effect of plotting only the area of concern. In this way a very detailed graph can be obtained.

Tables 8.1-8.4 provide a listing of program device type numbers to the actual device type and size.

Table 8.5 details what is contained in the various disk locations created by the programs.

To coordinate the devices in Figure 8.1, the procedure begins by entering element data using INPUT.F4. The program asks if data is to be read from disk or from the keyboard. The first element must be entered using the keyboard.

The proper procedure is to group the elements to be entered by wire or cable type as the program only handles one type of wire or cable at a time. When completed with one type, answer zero for all other questions and the computer writes the element data that it has computed on disk in the file RELA.DAT and stops. The new wire type polynomial coefficients are copied into RES.DAT and the program is ready to be run for the new wire type.

The cable coefficients used in the test problem are for mine power cable [5]. The coefficients are listed in Table 8.6.

After selecting keyboard, the base power and volts are entered. The computer then informs the user to begin element data entry. The data format (Chapter 3) is listed to aid the user. For example, the element from bus one to two of Figure 8.1 is entered as follows: 1/2/3000/6. For the element containing the 1 MVA transformer the following is input: 2/3/35/6/.001.005/1/15/4.1. The bus base voltages are

Table 8.2. Current-limiting fuses.

Program Type #	Limiting Fuses
1	5E
2	7E
3	10E
4	15E
5	20E
6	25E
7	30E
8	40E
9	50E
10	65E
11	80E
12	100E
13	125E
14	150E
15	200E

Table 8.3. Boric fuses.

Program Type #	Size
16	15E
17	20E
18	25E
19	30E
20	40E
21	50E
22	65E
23	80E
24	100E
25	125E
26	150E
27	175E
28	200E
29	250E
30	300E
31	400E
32	2-250E
33	2-300E
34	2-400E

Table 8.4. Molded-case circuit breakers.

Program #	Size
34	300 amps
35	350 amps
36	400 amps
37	450 amps
38	500 amps
39	600 amps
40	700 amps
41	800 amps
42	1000 amps
43	1200 amps

Table 8.5. Disc files.

DIGIT.F4:	Curve-smoothing program
DEV.F4:	Numonics input program
INPUT.F4	Element data-input program
LDFLO.F4:	Load-flow program
ZBUS.F4:	Formation of $ZBUS^{012}$ program
FALT.F4:	Solution of fault-current program
COORD.F4:	Coordinate device program
PLOTD.F4:	Plot device curves
RES.DAT:	Contains X/R line coefficients
TEST.DAT:	Contains element and load data in per unit
RELA.DAT:	Contains element and bus voltage data
XFOR.DAT:	Contains transformer inrush and withstand values
LOCUR.DAT:	Contains load currents
DEV.DAT:	Contains relay coefficients
DSK.XFR:	Contains load-flow currents in P.U.
DSK.DAT:	Contains load-flow analysis data
ZBUS.DAT:	Contains $ZBUS^{012}$
ZBUS.FOR:	Contains $ZBUS^{012}$ before loads are added
MUT.CO:	Contains mutual coupling data
FALT.DAT:	Contains per-unit fault currents
EFLT.DAT:	Contains per-unit fault voltages
LCUR.DAT:	Contains real values of load-flow currents
FCUR.DAT:	Contains real values of fault currents
ADEV.DAT:	Contains coordinated-device listing

Table 8.6. Mine power cable polynomial coefficients.

	R-resistance	X-reactance
a_0	0.758690526734D+01	0.6731452381407621D-01
a_1	- .1108155100578006D+02	- .1408237729509665D-01
a_2	0.6290760668063740D+01	- .1721580927352306D-02
a_3	- .1627214281565102D+01	.6845126542245339D-03
a_4	0.1604746751660286D+00	0.0

input next with the voltage in volts. Upon completion of entry of all bus base voltages, the computer types the computed element value in per unit. The load data is entered next. The program types the data entry format for the user. The 50 hp motor is entered as: 8/50/.85. The only other data to be entered are mutual coupling and zero sequence load impedance in ZBUS.F4 and fault impedance in FALT.F4. The sample problem uses no coupling and a fault impedance of zero. COORD.F4 is run next. The device type desired is entered when asked for by the program. A sample data input is given in Appendix D.

```
The following is a list of load locations and
currents.

Bus number  Load current

      4      84.0396680
      4      84.0396680
      7     293.6613400
      8      48.9435570


The following is the current flow in the system
under full load conditions as found from the
load flow analysis.

Bus    P     to   Bus    Q    Current at P  Current at Q

       1                 2      34.2226440     34.2226440
       2                 3      34.2226630    125.2048700
       3                 4      88.0372570     88.0372570
       3                 5      39.1722710     39.1722710
       5                 6      39.1721210    365.0129500
       6                 7     313.4835700    313.4835700
       6                 8      51.6573760     51.6573760


The following is the computed fault currents for a
3 phase fault and a line to ground fault.

  Flt Bs P      Element      3 Phase       Ln to Gnd

       2               1   9380.7892000   7053.9298000
       2               2    261.6855700    183.7685400
       2               3    677.8190200    595.4521200
       2               4    280.0413400    187.2686100
       2               5    280.3767000    186.2585700
       2               6   2226.3547000   1496.4606000
       2               7    383.7966800    270.9381000
       3               1   1247.2941000    833.3620700
       3               2   1247.2997000    826.2558200
       3               3    846.6528100    699.4013200
```

```
Flt Bs P    Element      3 Phase        Ln to Gnd
     3           4     349.8265800    206.0435600
     3           5     349.7991000    204.2971400
     3           6    2780.8159000   1648.7648000
     3           7     479.3659400    302.5682200
     4           1    1017.3596000    469.6438400
     4           2    1017.3546000    466.3491300
     4           3    4003.0938000   2210.2807000
     4           4     285.3288800    121.9024600
     4           5     285.3382800    121.3072000
     4           6    2268.1567000    972.7699300
     4           7     391.0159800    174.0103800
     5           1     881.8973500    308.3295700
     5           2     881.8899900    307.2671700
     5           3     598.6094700    260.1660700
     5           4    3803.2506000   1384.1725000
     5           5     353.1902700    108.5970700
     5           6    2807.5742000    901.2339600
     5           7     483.9912400    183.8482000
     6           1     791.2624200    148.9287400
     6           2     791.2645000    148.4773900
     6           3     537.0808000    125.4932500
     6           4    3416.9009000    668.5896000
     6           5    3416.8793000    569.4760500
     6           6    2839.6040000    517.3090900
     6           7     489.5244300    124.5511200
     7           1     350.4617800    103.1131400
     7           2     350.4624900    103.1008700
     7           3     237.8819000     70.4041400
     7           4    1513.3939000    445.7008900
     7           5    1513.3895000    443.9770900
     7           6   14273.8630000   4235.3970000
     7           7     216.8161500     64.2673200
     8           1     156.6019600     23.5717640
     8           2     156.6021300     23.5401980
     8           3     106.3009100     16.4766590
     8           4     676.2499600    102.2285800
     8           5     676.2485300    100.1874100
     8           6     561.9951900     84.0780670
     8           7    6769.2755000   1059.3405000
```

Using the device listing of ADEV.DAT and the plotted curves, it is possible to obtain all device settings.

Device number one is a molded-case circuit breaker. Its size is 300 A and its magnetic trip is on minimum setting.

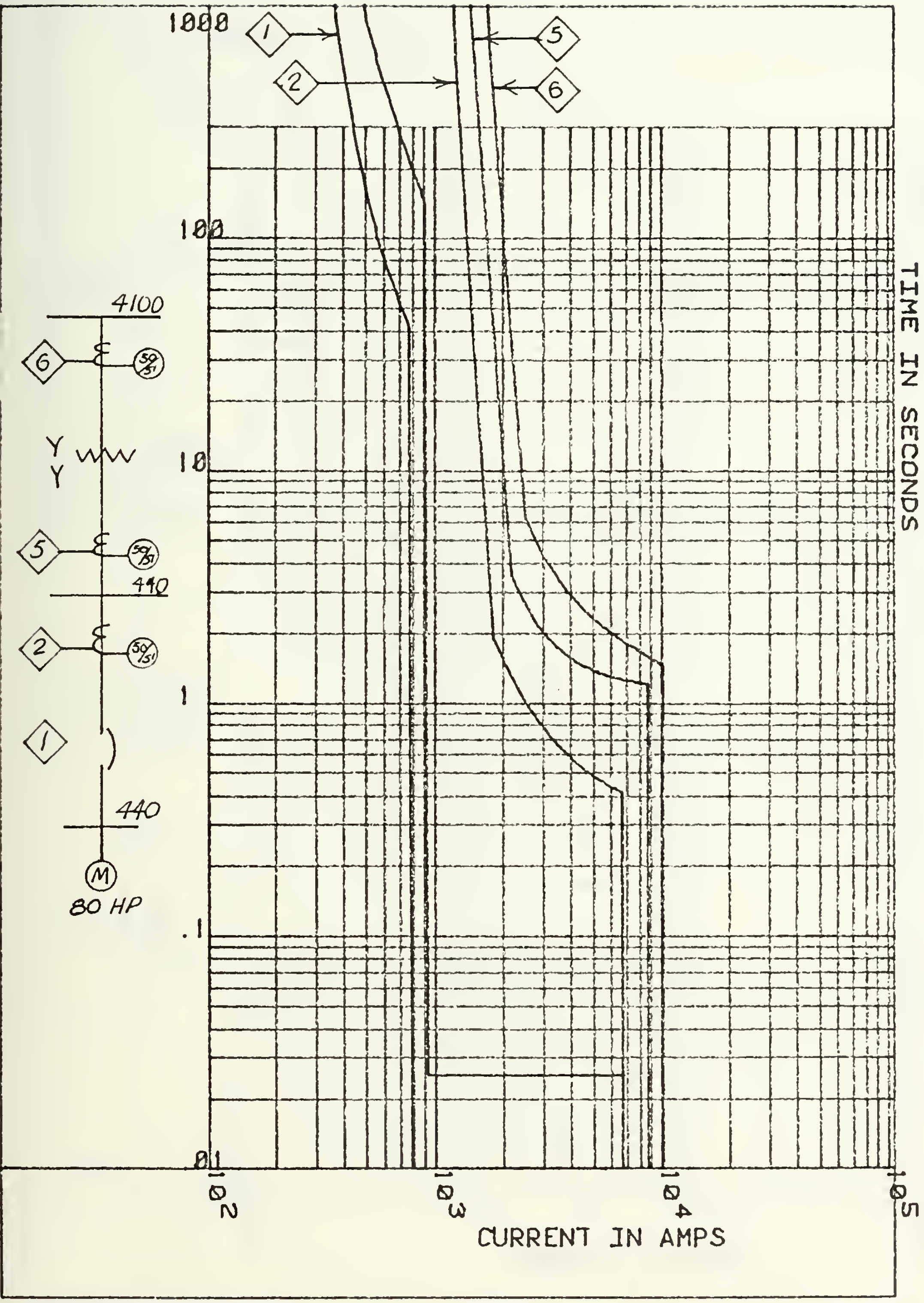
Device number 10 is a current-limiting fuse number 10.

Device number 12 is a solid-material high-voltage boric fuse number 24.

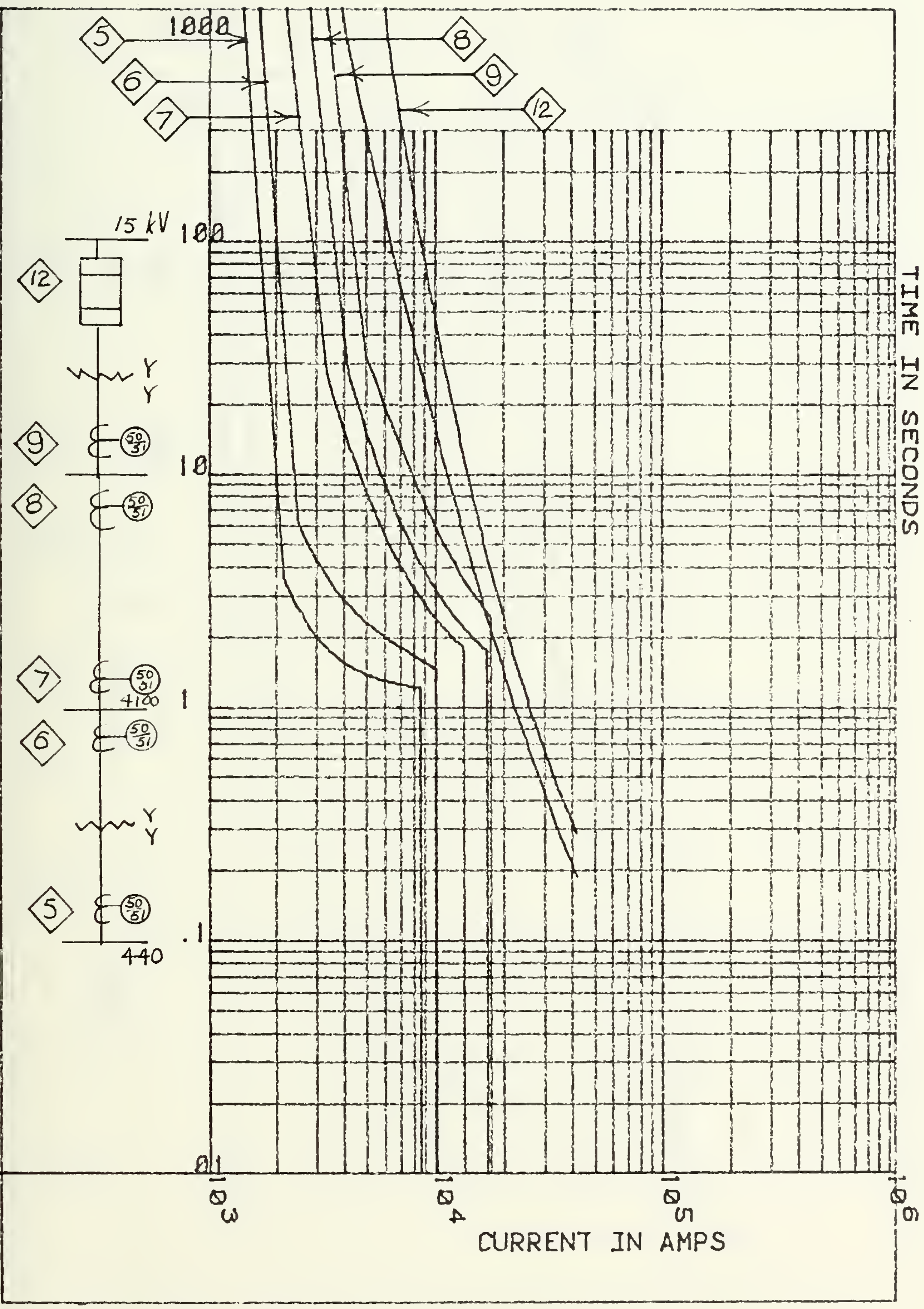
The relay data follows:

Device Number	Type Number	Ct	Tap	Time Dial	Magnetic
3	1	3 (120)	7	1/2	---
4	1	3	8	8	6
5	3	3	14	5	6
6	4	3	3	4	6
7	5	3	4	6	6
8	5	3	5	6	6
9	6	3	6	10	5.2
2	1	3	13	5	5.64
11	1	3	4	10	6
13	7	3	7	2	6
14	2	3	8	1/2	6

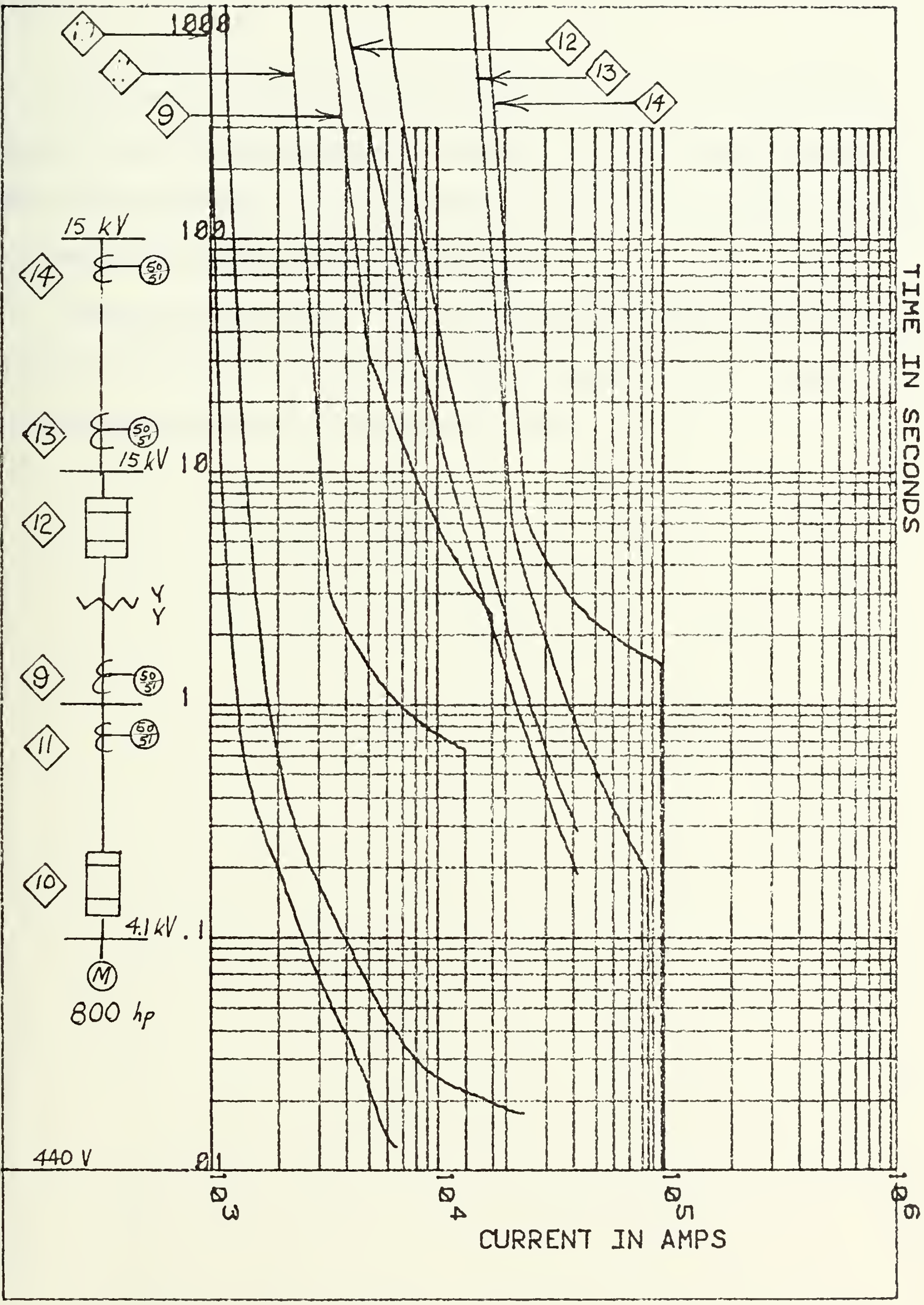
The magnetic trip of device number 9 had to be reduced after coordination to allow the device 12 to set properly to provide protection for the 1 MVA transformer. The graphs that follow show the devices plotted to the lowest voltage in the system, 440 V. All the graphs were computer generated. The first graph shows how a one-line computer diagram would be placed to aid in understanding the graphs.



igure 8.2. Example coordination plot.



gure 8.2. Continued.



gure 8.2. Continued.

Summary

The sample distribution system of Figure 8.1 was coordinated using the coordination programs. Using the device-plotting program PLOTD.F4 this coordination was checked to see if the computer generated coordination was satisfactory to the user.

During the research for this thesis, many observations were made and many features could be added to these programs in the future. This will be discussed in the chapter that follows.

9. CONCLUSIONS

In this thesis the author set out to represent a one-line distribution system in a per-unit system and to solve for three-phase and line-to-ground faults and to perform a load-flow analysis. The various protective devices-response curves needed to be represented by polynomial functions. These requirements were fulfilled using the methods detailed in Chapters 2-7.

The last task was to develop a computer procedure to coordinate a distribution system's protective devices. Fuses, molded-case circuit breakers, and relays were coordinated and their coordination graphs were plotted. There are many extensions for further work that could be done. These extensions will be discussed later.

While working with the various programs, several observations have proved useful when performing coordination. Coordination should not be an afterthought in designing a distribution system. It is easy to design a system that is impossible to coordinate correctly. A radial system is easier to coordinate. Moving from a load toward the source, the time-current curves shift to higher current-time responses so that when finally arriving at a transformer, providing proper transformer protection and coordination simultaneously may be impossible.

Coordination of the thermal-tripping elements of molded case circuit breakers is very difficult due to the difference between the minimum and maximum tripping time curves. Coordination of these

devices is best done manually with some overlap of the thermal time curves. The magnetic elements could then be adjusted to provide proper fault coordination. Those manually adjusted devices can be added to ADEV.DAT and be checked using PLOT.F4.

In systems with different voltages, the transformers should be of different sizes so that the smaller sizes are on the load side. This will allow proper coordination and protection. Relays are preferable to other types of protection on the high-voltage parts of the system in that their response can be controlled.

Recommendations for Further Work

1. Using an analog computer and the programs presented here, an entire distribution and generation system could be looked at in detail. A transient-stability study would show the effect of various device settings.

2. The INPUT.F4 program could be enlarged to automatically account for different transformer configurations.

3. Additional devices could be added easily to the program. The program allows up to twenty different relay types. The additional coefficient data is merely entered in disk DEV.DAT. Additional molded-case circuit breakers would be slightly more difficult in that it would require some program modification.

4. The plotting program could be enlarged to include drawing a one-line diagram in the space provided and adding other data directly on the graph.

5. The program could be expanded for the case of having one device in an element.

REFERENCES

1. Wagner, J. K., "Interactive Mine Power System Relay Coordination," M.S. Thesis, The Pennsylvania State University, 1980.

2. Radke, G. E., "A Method for Calculating Time Overcurrent Relay Settings by Digital Computer," IEEE Transactions on Power Apparatus and Systems, Vol. PAS-82 Special Supplement (1965): 189-205.

3. Albredt, R. E., Nisja, M. J., Feero, W. E., Rockeffeller, G. D., and Wagner, C. L., "Digital Computer Protective Device Coordination Program I--General Program Description," IEEE Transactions on Power Apparatus and Systems, Vol. PAS-83 (April 1964): 402-410.

4. Langhans, J. D. and Ronat, A. E., "Protective Devices Coordination via Computer Graphics," Proceedings of IEE Industry Applications Society Annual Meeting, October 1979, pp. 1209-1217.

5. Mining Cable Engineering Handbook, The Anaconda Company, 1977.

6. "Engineering Dependable Protection for an Electrical Distribution System," Bussman Manufacturing Division, McGraw-Edison Company, 1968.

7. Wagner, C. F., Evans, R. D., Symetrical Components, New York, McGraw-Hill, 1975.

8. Stevenson, W. D., Elements of Power System Analysis, New York, McGraw-Hill, 1933.

9. "NUMONICS Electronic Graphics Calculator," The Pennsylvania State University, Hybrid Computer Laboratory, 1976.

10. "Scientific Subroutine Package," Function "MINV," White Plains, New York, International Business Machines Company.

11. IEEE Recommended Practice for Protection and Coordination of Industrial and Commercial Power Systems. IEEE, 1975.

12. Stagg, G. W. and El-Abiad, A. H., Computer Methods in Power System Analysis, New York, McGraw-Hill, 1968.

13. Morley, L. A., Mine Power Systems, Vol. I, II, Bureau of Mines, 1981.

14. "Type Co (HiLo) Overcurrent Relay," Westinghouse I.L. 41-100B, 1976.

15. "Molded Case Circuit Breaker Ges G111B," General Electric.

APPENDIX A

CURVE-SMOOTHING PROGRAMS

```
C       TITLE:DIGIT.F4;CREATES FILES FROM DATA
         USING NUMONICS ARM
****************************************************
1       TYPE 5
5       FORMAT(' INPUT DEVICE NR..')
        ACCEPT 8,IDEV
8       FORMAT(I)
        OPEN(UNIT=IDEV)
        TYPE 10
10      FORMAT(' INPUT X-AXIS DIMENSION,Y-AXIS
         DIMENSION'//'..,')
        ACCEPT 15,XAX,YAX
15      FORMAT(2F)
        TYPE 20
20      FORMAT(' INPUT LENGTH PER LOG10;X,Y..')
30      ACCEPT 35,XL,YL
35      FORMAT(2F)
        XAX=100*XAX/XL
        YAX=100*YAX/YL
        CALL OPEN(30,'DEVICE','NUM','MODE','IMAGE',
        1'ACCESS','SEQIN','BUFFERS',4)
        CALL NUMRES(30,4)
        CALL LOCK(1,1)
100     READ(30,END=200) IDATA
        IX=(IDATA/"1000000).AND."37777
        IY=IDATA.AND."37777
        IFLAGS=(IDATA.AND."740000)/"40000
        X=IX/XAX
        Y=IY/YAX
        WRITE(IDEV,150)IFLAGS,X,Y
150     FORMAT(I,2F)
        GO TO 100
200     CALL NUMCLO
        CALL UNLOCK(1,1)
        CLOSE(UNIT=IDEV)
        TYPE 210
210     FORMAT(' CONTINUE?..,'$)
        ACCEPT 220,IC
220     FORMAT(I)
        IF(IC.EQ.0)GO TO 1
        STOP
        END
```

```
**************************************************
C       CURVE FITTING PROGRAM:DEV.F4
**************************************************
        INTEGER L(7),MM(7)
        REAL X(100),YS(100)
        REAL*8 YM(100),G(7,7),U(7),A(0/6),GG(7,7),ERROR(100)
        TYPE 100
100     FORMAT(1X,'CHOOSE DIMENSION OF FIT  '$)
        ACCEPT 110,ND
110     FORMAT(I)
        TYPE 120
120     FORMAT(1X,'CHOOSE NO. OF POINTS   '$)
        ACCEPT 110,M
        TYPE 130
130     FORMAT(/'CHOOSE INPUT FILES'/' ...,    '$)
        ACCEPT 110,IUNIT1
        DO 140  I=1,M
        READ(IUNIT1,135)J,X(I),YS(I)
135     FORMAT(I,2F)
        YS(I)=ALOG10(YS(I))
140     CONTINUE
        DO 10 K=1,M
        DO 10 I=1,ND
        U(I)=U(I)+YS(K)*X(K)**(I-1)
        DO 10 J=1,ND
        G(I,J)=G(I,J)+X(K)**(I+J-2)
10      CONTINUE
        CALL DMINV(G,7,ND)
        DO 20 I=0,ND-1
        DO 20 J=1,ND
        A(I)=A(I)+G(I+1,J)*U(J)
20      CONTINUE
        DO 30 I=1,M
        DO 30 J=0,ND-1
        YM(I)=YM(I)+A(J)*X(I)**J
30      CONTINUE
        DO 40 I=1,M
40      SQUERR=SQUERR+(YS(I)-YM(I))**2
        TYPE 160,SQUERR
160     FORMAT(' THE INTERGRAL-SQUARED ERROR IS:    ',E)
        OPEN(UNIT=40,FILE='A.DAT')
        WRITE(40,170)A
        CLOSE(UNIT=40)
170     FORMAT(D)
        TYPE 99,IUNIT1
99      FORMAT(' IUNIT=',I)
        DO 50 I=1,M
50      ERROR(I)=DABS((YM(I)-YS(I))/YS(I))
        OPEN(UNIT=41,FILE='YINOUT.DAT')
        WRITE(41,180)(X(I),YS(I),YM(I),ERROR(I),I=1,M)
180     FORMAT(2F,2D)
```

```
        CLOSE(UNIT=41)
        STOP
        END
C       TITLE:DMINV;SUBROUTINE TO INVERT A REAL SQUARE
MATRIX DOUBLE
C       PRECISION
        SUBROUTINE DMINV(A,MDIM,N)
        REAL*8 A(MDIM,N)
        INTEGER L(100),M(100)
C
C       CONVERTED FROM SSP ROUTNE MINV
C
        DOUBLE PRECISION BIGA,HOLD
        DO 80 K=1,N
        L(K)=K
        M(K)=K
        BIGA=A(K,K)
        DO 20 J=K,N
        DO 20 I=K,N
10      IF(DABS(BIGA)-DABS(A(I,J))) 15,20,20
15      BIGA=A(I,J)
        L(K)=I
        M(K)=J
20      CONTINUE
        J=L(K)
        IF(J-K) 35,35,25
25      DO 30 I=1,N
        HOLD=-A(K,I)
        A(K,I)=A(J,I)
30      A(J,I)=HOLD
35      I=M(K)
        IF(I-K) 45,45,38
38      DO 40 J=1,N
        HOLD=-A(J,K)
        A(J,K)=A(J,I)
40      A(J,I)=HOLD
45      DO 55 I=1,N
        IF(I-K) 50,55,50
50      A(I,K)=A(I,K)/(-BIGA)
55      CONTINUE
        DO 65 I=1,N
        HOLD=A(I,K)
        DO 65 J=1,N
        IF(I-K) 60,65,60
60      IF(J-K) 62,65,62
62      A(I,J)=HOLD*A(K,J)+A(I,J)
65      CONTINUE
        DO 75 J=1,N
        IF(J-K) 70,75,70
70      A(K,J)=A(K,J)/BIGA
75      CONTINUE
```

```
          A(K,K)=(1.0000000)/BIGA
80        CONTINUE
          K=N
100       K=K-1
          IF(K) 150,150,105
105       I=L(K)
          IF(I-K) 120,120,108
108       DO 110 J=1,N
          HOLD=A(J,K)
          A(J,K)=-A(J,I)
110       A(J,I)=HOLD
120       J=M(K)
          IF(J-K) 100,100,125
125       DO 130 I=1,N
          HOLD=A(K,I)
          A(K,I)=-A(J,I)
130       A(J,I)=HOLD
          GO TO 100
150       RETURN
          END
```

APPENDIX B

COORDINATION AND PLOTTING PROGRAMS

```
*******************************************************
C       INPUT.IT INPUTS DATA FROM A TERMINAL,PLACES IN
C        THE PROPER FORMAT FOR OTHER PROGRAMS
*******************************************************
C
C
        DIMENSION
L(30,30,2),PU(30,30,1),RES(30,30,2),E(30),XL(30,30,2)
        DIMENSION
S(300),ZER(30,30,2),RC(7),XC(7),ZG(30),XD(30)
        COMPLEX T1,TPU,PU,S,S1,PU1,PZ,CUR,ZER,ZG
        REAL*8 A(7),AA(7)
        REAL L
**********************************************************
C       IF THE PROGRAM HAS ALREADY BEEN RUN ONCE ALL DATA
C       NEED NOT BE ENTERED AGAIN
***********************************************************
        TYPE 3
3       FORMAT(' DO YOU WANT TO READ DATA FROM A FILE OR
FROM KYBRD?'/
        2' <CR>FOR INPUT FROM TTY,1 FOR READ...')
        ACCEPT 11,I
        IF(I.NE.0)GO TO 400
************************************************************
C       INPUT OF DATA ON LINES
********************************************************
        TYPE 5
5       FORMAT(2X,'ENTER THE PROBLEM BASE IN MVA')
        ACCEPT 10,ZBASE
        TYPE 6
6       FORMAT(2X,'ENTER THE PROBLEM BASE IN KV')
        ACCEPT 10,EBASE
10      FORMAT(F)
11      FORMAT(I)
        GO TO 14
*********************************************************
C       PROGRAM RETURNS HERE AFTER READING DATA AT 400
********************************************************
12      TYPE 13
13      FORMAT(' OLD DATA IS ENTERED,READY FOR NEW LINE
DATA..')
        ACCEPT 11,I
14      TYPE 15
15      FORMAT(/,2X,'ENTER ELEMENT DATA AS
FOLLOWS:BUS()TO()(LENGTH)
        1(SP)SIZE',/,2X,'(TRANSFORMER
IMPEDENCE(P.U.))(TRANSFORMER SIZE
        2IN MVA) SP',/, 2X,'(TRANSFORMER HIGH VOLTAGE IN KV)
SP',/,'
        3(TRANSFORMER LOW VOLTAGE IN KV]...,'/)
********************************************************
```

```
C       THESE TWO FILES CONTAIN COEFFICIENT DATA FO 40 BELOW
C       THEY ARE CREATED USING THE CURVE FITTING
PROGRAM:CURV.F4
**********************************************************
        OPEN(UNIT=40,FILE='RES.DAT')
        OPEN(UNIT=35,FILE='XFOR.DAT')
        READ(40,20)(A(I),I=1,7)
        READ(40,20)(AA(I),I=1,7)
20      FORMAT(D)
        DO 21 I=1,7
        RC(I)=A(I)
        XC(I)=AA(I)
21      CONTINUE
        CLOSE(UNIT=40)
        CLOSE(UNIT=41)
22      ACCEPT 24,I,J,XLL,WIR,I1,FS1,TVH,TVL
24      FORMAT(2I,7F)
        IF(I.EQ.0) GO TO 110
        TYPE 26
        JJ=1
26      FORMAT('  PARALLEL LINE   1 TO 3...',$)
        ACCEPT 27,JJ
27      FORMAT(I)
        IF(JJ.EQ.0)JJ=1
        L(I,J,JJ)=XLL
        IF(WIR.EQ.140)GO TO 31
        IF(WIR.EQ.130)GO TO 32
        IF(WIR.EQ.120)GO TO 33
        IF(WIR.EQ.110)GO TO 34
        SA=WIR
        SB=WIR
        IF(SA.LT.37)GO TO 35
        GO TO 40
31      SA=-3
        GO TO 35
32      SA=-2
        GO TO 35
33      SA=-1
        GO TO 35
34      SA=0
        GO TO 35
*****************************************************
C       CONVERSION FROM AWG TO MCM
*****************************************************
35      SC=1.1229**(36-SA)
        SD=.005*SC
        SE=SD**2
        SB=1000*SE
********************************************************
C       FORMULAE FOR LINE RESISTANCE AND REACTANCE
*******************************************************
```

```
40
RES(I,J,JJ)=RC(1)+RC(2)*(ALOG10(SB))+RC(3)*(ALOG10(SB
)**2)+

1RC(4)*(ALOG10(SB)**3)+RC(5)*(ALOG10(SB)**4)+RC(6)*(A
LOG10(SB)
        2**5)+RC(7)*(ALOG10(SB)**6)

XL(I,J,JJ)=XC(1)+XC(2)*(ALOG10(SB))+XC(3)*(ALOG10(SB)
**2)+

1XC(4)*(ALOG10(SB)**3)+XC(5)*(ALOG10(SB)**4)+RC(6)*(A
LOG10(SB)
        2**5)+XC(7)*(ALOG10(SB)**6)
        TYPE 50,RES(I,J,JJ),SA,SB,XL(I,J,JJ)
50      FORMAT(2X,'LINE RES. ',3F,/,'LINE REACT..',F)
        IF(CABS(T1).EQ.0)GO TO 85
        PU(I,J,JJ)=T1*(ZBASE/TS1)*((TVH/EBASE)**2)
        TYPE 55,I,J
55      FORMAT(' INPUT TRANSFORMER ZN IN P.U. REFER TO BUS
SIDE,(BUS
        1',I3,')SP(BUS ',I3,')..,')
        ACCEPT 60,S1,PU1
60      FORMAT(4F)
        TV=TVH*1E3
        ZG(I)=S1*(ZBASE/TS1)*((EBASE/TVH)**2)
        ZG(J)=PU1*(ZBASE/TS1)*((EBASE/TVL)**2)
        CURB=TS1*1E6/TV
        TINRSH=12*CURB
        TWST1=25*CURB
        TWST2=14.4*CURB
        WRITE(35,70)I,J,TINRSH,TWST1,TWST2,TS1
70      FORMAT(2I,4F)
85      IF(IMAX.GT.I) GO TO 90
        IMAX=I
90      IF(JMAX.GT.J) GO TO 100
        JMAX=J
100     ZER(I,J,JJ)=CMPLX(0.0,XL(I,J,JJ))
        TYPE 102,ZER(I,J,JJ),SS1
102     FORMAT(3F)
        GO TO 22
110     OPEN(UNIT=31,FILE='RELA.DAT')
        OPEN(UNIT=32,FILE='TEST.DAT')
        OPEN(UNIT=33,FILE='LOCUR.DAT')
        IF(IMAX.LT.JMAX) GO TO 115
        IB=IMAX
        GO TO 118
115     IB=JMAX
**********************************************************
C       BUS VOLTAGES ARE INPUT
**********************************************************
```

```
118      DO 120 I=1,IB
         TYPE 119,I
119      FORMAT(' INPUT BUS',I3,'VOLTAGE...,',$)
         ACCEPT 10,E2
         IF(E2.EQ.0)GO TO 121
         E(I)=E2
120      CONTINUE
121      TYPE 122
122      FORMAT(/,2X,'DATA IS ACCEPTED AND A LIST FOLLOWS')
         TYPE 125,ZBASE
125      FORMAT(2X,'PROBLEM BASE IN MVA..,',F)
         WRITE(31,126)ZBASE,EBASE,IB
126      FORMAT(2F,I)
         E1=E(1)/(EBASE*1E3)
         WRITE(32,128),IB,E1
128      FORMAT(I,2F)
         DO 135 I=1,30
         DO 133 J=1,30
         DO 133 JJ=1,2
         IF(L(I,J,JJ).EQ.0)GO TO 133
         NN=NN+1
         PZ=(CMPLX(RES(I,J,JJ),XL(I,J,JJ)))*(.001*L(I,J,JJ))
         PZZ=(ZBASE*1E6)/(E(I)**2)
         PZ=PZ*PZZ
         T1=PZ+PU(I,J,JJ)
         ZER(I,J,JJ)=ZER(I,J,JJ)*PZZ*(.001*L(I,J,JJ))+T1
         TYPE 130,NN,L(I,J,JJ),T1
130      FORMAT(2X,'ELEMENT  ',I3,5X,'LENGTH',F,/,
         11X,'TRANSFORMERPER UNIT IMPEDENCE PLUS LINE
IMPEDENCE',2F)

WRITE(31,131)I,J,JJ,L(I,J,JJ),RES(I,J,JJ),PU(I,J,JJ),
XL(I,J,JJ)
         1,ZG(I),ZG(J)
131      FORMAT(3I,9F)
         WRITE(32,132)NN,I,J,JJ,T1,ZER(I,J,JJ),ZG(I),ZG(J)
132      FORMAT(4I,8F)
133      CONTINUE
135      CONTINUE
         WRITE(31,136)
         WRITE(32,136)
136      FORMAT('  0 0 0 0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0')
         DO 138 I=1,IB
         WRITE(31,137)I,E(I)
137      FORMAT(I,2F)
138      CONTINUE
         WRITE(31,139)
139      FORMAT(' 0 0.0 0.0')
         TYPE 140,IB
140      FORMAT(10X,'NUMBER OF BUSSES IS...,',I)
145      TYPE 150
```

```
**********************************************************
150     FORMAT(//' NOW ENTER THE LOAD DATA IN THE FOLLOWING
MANNER;...
        1'/'(BUS  )(LOAD IN KVA)(PWR FACTOR)...OR...
        2'/' (BUS ) (LOAD IN HP)(PWR FACTOR)....OR....'/
        3'(BUS )(LOAD IN AMPS)(PWR FACTOR)')
        DO 300 N=1,IB
                TYPE 160
160             FORMAT(' WHAT METHOD WILL YOU USE
?..VA,HP,OR AMPS??')
                ACCEPT 162,MD
162             FORMAT(A4)
                TYPE 164,MD
164             FORMAT(2X,A4,' ENTER...')
                IF(MD.EQ.'HP1') GO TO 170
                IF(MD.EQ.'HP') GO TO 170
                IF(MD.EQ.'AMPS') GO TO 180
                IF(MD.EQ.'S') GO TO 305
                MD='VA'
                ACCEPT 165, J,AS1,PFAC
165             FORMAT(I,2F)
                SZTZ=AS1/(ZBASE*1E3)
                SZR=SZTZ*PFAC
                PCOS=ACOS(PFAC)
                SZI=SZTZ*SIN(PCOS)
                S(J)=CMPLX(SZR,SZI)
                AI=AS1*1E3/(E(J)*SQRT(3.0))
                WRITE(33,168)J,AI
168             FORMAT(I,F)
                ST=MD
                GO TO 300
170             ACCEPT 175,I,PRH,PFAC
175             FORMAT(I,3F)
                SZTZ=PRH*746
                SZR=SZTZ*PFAC
                PCOS=ACOS(PFAC)
                SZI=SZTZ*SIN(PCOS)
                S(I)=CMPLX(SZR,SZI)/(ZBASE*1E6)
                IF(MD.EQ.'HP')XD(I)=.17/CABS(S(I))
                AI=SZTZ/(SQRT(3.0)*E(I))
                WRITE(33,178)I,AI
178             FORMAT(I,F)
                ST=MD
                GO TO 300
180             ACCEPT 185,I,CURT,PFAC
185             FORMAT(I,3F)
                WRITE(33,187)I,CURT
187             FORMAT(I,F)
                CURR=CURT*PFAC
                PCOS=ACOS(PFAC)
                CURI=CURT*SIN(PCOS)
```

```
                CUR=CMPLX(CURR,CURI)
                CUR=CUR*(SQRT(3.0)*EBASE/ZBASE)
                S(I)=3*EBASE*CONJG(CUR)/SQRT(3.0)
                ST=MD
300     CONTINUE
305     DO 320 N=1,IB
                NN=N
                TYPE 310,N,S(N),MD
310     FORMAT(' BUS ',I3,2F,A4,2F)
                WRITE(31,312)NN,S(NN)
312     FORMAT(I,2F)
                WRITE(32,315)N,S(N),XD(NN)
315     FORMAT(I,3F)
320     CONTINUE
        WRITE(31,330)
        WRITE(32,331)
330     FORMAT('  0 0.0 0.0')
331     FORMAT('  0 0.0 0.0')
        WRITE(33,332)
332     FORMAT(' 0 0.0')
        WRITE(35,335)
335     FORMAT(' 0 0 0.0 0.0 0.0 0.0 ')
        GO TO 900
400     OPEN(UNIT=36,FILE='RELA.DAT')
        READ(36,411) ZBASE,EBASE,IB
        IMAX=IB
411     FORMAT(2F,I)
450     READ(36,456)I,J,JJ,XL3,RES1,PU1,XL2,T1,S1
456     FORMAT(3I,9F)
        IF(I.EQ.0)GO TO 470
        L(I,J,JJ)=XL3
        RES(I,J,JJ)=RES1
        PU(I,J,1)=PU1
        XL(I,J,JJ)=XL2
        ZER(I,J,JJ)=CMPLX(0.0,XL2)
        ZG(I)=T1
        ZG(J)=S1
        IF(I.NE.0)GO TO 450
470     READ(36,472)I,E(I)
472     FORMAT(I,2F)
        IF(I.NE.0) GO TO 470
457     READ(36,458)I,S1
458     FORMAT(I,2F)
        S(I)=S1
        IF(I.NE.0) GO TO 457
        REWIND 36
        JJ=0
        CLOSE(UNIT=36)
        GO TO 12
900     CLOSE(UNIT=31)
        CLOSE(UNIT=32)
```

```
        CLOSE(UNIT=35)
        STOP
        END
***********************************************************
C       LDFLO.F4:A LOAD FLOW PROGRAM
**********************************************************
        DIMENSION
E(33),Y1(33,33),PSCH(33),QSCH(33),PCAL(33),

1QCAL(33),PDIFF(33),QDIFF(33),IC(33),S(33),JAC(68,68)
,
        2PQVEC(33),EFDV(64),QQVEC(33),
        3YS(33,33),Y(33,33),CRFLO(33,33),U(68),V(68),
        4EDIFF(33)
        INTEGER P,Q,M,U,V,PP
        COMPLEX E,IC,Y,Z,YS,Y1,PFLO,PLOS,YSH,EDIFF,
        1S,CRFLO,PFLO1,ZG1,ZNI,ZNJ
        EQUIVALENCE (CRFLO,Y1)
        REAL JAC
        DIMENSION EIDV(64)

OPEN(UNIT=30,DEVICE='DSK',FILE='TEST.DAT',DISPOSE='SA
VE')
        OPEN(UNIT=33,DEVICE='DSK',FILE='DSK.XFR')
*********************************************************
CREAD VALUES OF IMPEDANCES  YSHUNT ADMITTANCES

C       888888*****IF SHUNT ADMITTANCES ARENOT IGNORED
REMOVE
***THE C IN THE LINE YS(Q,P)**************************
C******************************************************
        READ(30,2) IB,E1
2       FORMAT(I,F)
        E(1)=CMPLX(E1,0.0)
        DO 10 I=1,1000
        READ(30,5)NN,P,Q,JJ,Z,ZG1,ZNI,ZNJ
5       FORMAT(4I,8F)
        IF(P.EQ.0) GO TO 15
        YS(P,Q)=YSH
C       YS(Q,P)=YSH
        IF(CABS(Y1(P,Q)).NE.0)Z=1.0/(Y1(P,Q)+1/Z)
        Y1(P,Q)=1.0/Z
        Y1(Q,P)=Y1(P,Q)
        PRINT 6,P,Q,Y1(P,Q),YS(P,Q)
6       FORMAT(2I,4F)
10      CONTINUE
***********************************************************
CCOMPILE TERMS FOR YBUS
C******************************************************
15      DO 50 P=1,IB
        Y(P,P)=0
```

```
        DO 40 Q=1,IB
        IF(P.EQ.Q) GO TO 40
        Y(P,P)=Y(P,P)+Y1(P,Q)+YS(P,Q)
        Y(P,Q)=-Y1(P,Q)
40      CONTINUE
50      CONTINUE
C************************************************
C PRINT OUT VALUES OF YBUS
C**************************************************
        PRINT 51
51      FORMAT ('0',10X,'BUS IMPEDANCE MATRIX,YBUS')
        DO 58 I=1,IB
        PRINT 54,(Y(I,J),J=1,IB)
54      FORMAT('0',100F)
58      CONTINUE
C***************************************************
C READ VALUES OF POWER AT BUSSES
C***************************************************
        DO 60 P=1,IB
        READ(30,59) I,PS,QS
        IF(I.EQ.0)GO TO 62
        IF(I.EQ.1)GO TO 60
        PSCH(I)=-PS
        QSCH(I)=-QS
59      FORMAT(I,2F)
60      CONTINUE
C***************************************************
C ASSUME VALUES OF VOLTAGE AT BUSSES
C***************************************************
62      DO 70 P=2,IB
        E(P)=(1.0,0.0)
70      CONTINUE
        K=0
        DO 80 I=1,IB
                TYPE 75,I,E(I)
75      FORMAT(I,2F)
80      CONTINUE
122     PRINT 123,K
123     FORMAT ('0',85X,'ITERATION = ',I2)
C*************************************************
C CALCULATES REAL  REACTIVE BUSS POWERS
C*************************************************
        DO 150 P=2,IB
        PSUM=0
        QSUM=0
        DO 140 Q=1,IB

P1=REAL(E(P))*(REAL(E(Q))*REAL(Y(P,Q))-AIMAG(E(Q))*AI
MAG(Y

1(P,Q)))+AIMAG(E(P))*(AIMAG(E(Q))*REAL(Y(P,Q))+REAL(E
```

```
(Q))*
        2AIMAG(Y(P,Q)))

Q1=AIMAG(E(P))*(REAL(E(Q))*REAL(Y(P,Q))-AIMAG(E(Q))*A
IMAG(Y

1(P,Q)))-REAL(E(P))*(AIMAG(E(Q))*REAL(Y(P,Q))+REAL(E(
Q))*AI
        2MAG(Y(P,Q)))
        PSUM=PSUM+P1
        QSUM=QSUM+Q1
140     CONTINUE
        PCAL(P)=PSUM
        QCAL(P)=QSUM
        PRINT 145,PCAL(P),QCAL(P)
145     FORMAT(2F)
150     CONTINUE
C****************************************************
C PRINT OUT VALUES OF CALCULATED BUSS POWERS
C****************************************************
        PRINT 155,K
155     FORMAT('0','REAL AND REACTIVE BUSS POWERS K=',I2)
        DO 158 I=2,IB
        PRINT 157,PCAL(I),QCAL(I)
157     FORMAT ('0',2F)
158     CONTINUE
        M=0
        DO 160 P=2,IB
        EP=0.01
        PDIFF(P)=PSCH(P)-PCAL(P)
        QDIFF(P)=QSCH(P)-QCAL(P)
        TDIFF=ABS(PDIFF(P))+ABS(QDIFF(P))
        IF(TDIFF.LT.EP) GO TO 160
        M=1
160     CONTINUE
C*********************************************
C PRINT OUT VALUES OF DIFFERENTIAL BUSS POWERS
C***********************************************
        PRINT 165,K
165     FORMAT('0','DIFFERENTIAL PQ   K=',3X,I2)

        DO 168 I=2,IB
        PRINT 167,PDIFF(I),QDIFF(I)
167     FORMAT ('0',2F)
168     CONTINUE
        IF(M.EQ.0) GO TO 400
        DO 170 I=2,IB
        S(I)=CMPLX(PCAL(I),QCAL(I))
170     CONTINUE
C******************************************
C  CALCULATE BUSS CURRENTS I(P) EXCEPT AT SLACK BUS
```

```
C********************************************
         DO 175 I=2,IB
         IC(I)=CONJG(S(I))/CONJG(E(I))
         PRINT 173,IC(I)
173      FORMAT(F)
175      CONTINUE
C******************************************
C  CALCULATE JACOBIAN USING PARTITIONS 'J1,J2,J3,J4'
C******************************************
C  JACOBIAN 'J1'
C***************
         IBDJ=2*IB-2
         IBJ=IB-1
         DO 190 M=1,IBJ
         DO 180 N=1,IBJ
         P=M+1
         Q=N+1

JAC(M,N)=(REAL(E(P))*REAL(Y(P,Q)))+(AIMAG(E(P))*AIMAG
(Y(P,Q)))
         IF(P.NE.Q) GO TO 177
         JAC(M,N)=JAC(M,N)+REAL(IC(P))
177      CONTINUE
C        PRINT 178,JAC(M,N),E(P),Y(P,Q)
178      FORMAT(5F)
180      CONTINUE
190      CONTINUE
*************************************************************
C  JACOBIAN 'J2'
************************************************************
         DO 210 M=1,IBJ
         DO 200 N=IB,IBDJ
         P=M+1
         Q=N-IB+2

JAC(M,N)=-REAL(E(P))*AIMAG(Y(P,Q))+AIMAG(E(P))*REAL(Y
(P,Q))
         IF(P.NE.Q) GO TO 195
         JAC(M,N)=JAC(M,N)+AIMAG(IC(P))
195      CONTINUE
200      CONTINUE
210      CONTINUE
***********************************************************
C  JACOBIAN 'J3'
*********************************************************
         DO 230 M=IB,IBDJ
         DO 220 N=1,IBJ
         P=M-IB+2
         Q=N+1

JAC(M,N)=-REAL(E(P))*AIMAG(Y(P,Q))+AIMAG(E(P))*REAL(Y
```

```
(P,Q))
        IF(P.NE.Q) GO TO 215
        JAC(M,N)=JAC(M,N)-AIMAG(IC(P))
215     CONTINUE
220     CONTINUE
230     CONTINUE
**********************************************************
C  JACOBIAN 'J4'
*******************************************************
        DO 250 M=IB,IBDJ
        DO 240 N=IB,IBDJ
        P=M-IB+2
        Q=N-IB+2

JAC(M,N)=-AIMAG(E(P))*AIMAG(Y(P,Q))-REAL(E(P))*REAL(Y
(P,Q))
        IF(P.NE.Q) GO TO 235
        JAC(M,N)=JAC(M,N)+REAL(IC(P))
235     CONTINUE
240     CONTINUE
250     CONTINUE
        DO 334 M=1,IBDJ
C       PRINT 333,(JAC(M,N),N=1,IBDJ)
333     FORMAT(18F)
334     CONTINUE
*************************************************************
*************************************************************
C  COMPUTE INVERSE OF JACOBIAN
        CALL MINV(JAC,68,IBDJ)
**********************************************************
C       PRINT 344,(JAC(M,N),N=1,IBDJ)
344     FORMAT(18F)
345     CONTINUE
***********************************************************
C  COMPUTE DIFERENCE IN VOLTAGE USING JACOBIAN INVERSE 'JAC'
**********************************************************
        DO 346 I=2,IB
        M=I-1
        PQVEC(M)=PDIFF(I)
        QQVEC(M)=QDIFF(I)
346     CONTINUE
        J=IB-1
        DO 351 M=1,J
        EFDVD=0
        EFDQD=0
        EIDD=0
        EIQV=0
        DO 349 N=1,J
        EFDVD=EFDVD+JAC(M,N)*PQVEC(N)
349     CONTINUE
        DO 350  N=IB,IBDJ
```

```
       NQ=N-J
       EFDQD=EFDQD+JAC(M,N)*QQVEC(NQ)
350    CONTINUE
       EFDV(M)=EFDQD+EFDVD
351    CONTINUE
       DO 360 M=IB,IBDJ
       EIDD=0
       EIQV=0
       DO 358 N=1,J
       EIDD=EIDD+JAC(M,N)*PQVEC(N)
358    CONTINUE
       DO 359 N=IB,IBDJ
       NQ=N-J
       EIQV=EIQV+JAC(M,N)*QQVEC(NQ)
359    CONTINUE
       EIDV(M)=EIQV+EIDD
360    CONTINUE
*****************************************************
C  COMPUTE NEW BUS VOLTAGES USING 'EFDV' EFDIFFERENTIAL
VECTOR
****************************************************
       DO 370 P=2,IB
       M=P-1
       N=M+J
       IBZ=2*(IB-1)
       IF(N.GT.IBZ) GO TO 370
       EDIFF(P)=CMPLX(EFDV(M),EIDV(N))
       PRINT 368,P,EDIFF(P)
368    FORMAT(I,2F)
       E(P)=E(P)+EDIFF(P)
370    CONTINUE
       IF(K.EQ.10)GO TO 390
       K=K+1
       GO TO 122
390    TYPE 395
395    FORMAT(' AN ERROR IS MADE IN DATA, LOAD FLOW DID NOT
CONVERGE')
400    CONTINUE
500    CLOSE(UNIT=30)
*****************************************************
C      COMPUTE POWER FLOWS BUS P TO Q
C      COMPUTE CURRENT FLOWS AND POWER LOSSES
C      COMPUTE SLACK BUS POWER
*****************************************************
       OPEN(UNIT=31,FILE='DSK')
       S(1)=0
503    FORMAT(I)
       DO 520 P=1,IB
       DO 515 Q=1,IB
       IF(P.EQ.Q) GO TO 515
       IF(Y(P,Q).EQ.(0.0,0.0)) GO TO 515
```

```
        PP=P
***************************************************
C       IF YOU ARE NOT NEGLECT SHUNT ADMITT REMOVE THE NEXT
C
**********************************************************
        PFLO=CONJG(E(P))*((E(P)-E(Q))*Y1(P,Q))
C       1+(CONJG(E(P))*E(P)*YS(P,Q))
        PFLO1=CONJG(E(Q))*((E(Q)-E(P))*Y1(P,Q))
C       1+(CONJG(E(Q))*E(Q)*YS(P,Q))
        CRFLO(P,Q)=PFLO/CONJG(E(P))
        PLOS=PFLO+PFLO1
**********************************************************
C       TAKE THE CONJUGATE OF THE POWER FLOWS FOR FORMAT
P-JQ
**********************************************************
        PLOS=CONJG(PLOS)
        PFLO=CONJG(PFLO)
        IF(P.NE.1) GO TO 505
        S(1)=PFLO+S(1)
505     WRITE(31,509)P,Q,PFLO,CRFLO(P,Q),PLOS
        WRITE(31,510)PP,E(PP)
509     FORMAT(6X,'POWER FLOW
BUS',I2,'TO',I2,'IS',2F,2X,'CURRENT
        1FLOW IS',2F,'POWER LOSS IS',2F)
510     FORMAT(5X,'BUS',I2,'VOLTAGE IS',2F)
515     CONTINUE
520     CONTINUE
        WRITE(31,540) S(1)
540     FORMAT(6X,'POWER AT SLACK BUS 1 IS',2F)
        DO 550 M=1,IB
                DO 545 N=1,IB
                IF(CABS(CRFLO(M,N)).GT.0.000001)
WRITE(33,543)M,N,
        1CRFLO(M,N)
543     FORMAT(2I,2F)
545             CONTINUE
550     CONTINUE
        WRITE(33,555)
555     FORMAT(' 0   0   0.0 0.0')
        CLOSE(UNIT=33)
        DO 600 M=1,IB
        DO 580 N=1,IB
        S(2)=E(M)*CONJG(CRFLO(M,N))
        PFAD=CABS(S(2))
        IF(PFAD.EQ.0) GO TO 580
        PFAC=-(REAL(S(2)))/(PFAD)
        WRITE(31,570)M,N,PFAC
570     FORMAT(4X,'POWER FACTOR BUS',I3,'TO',I3,'IS',F)
580     CONTINUE
600     CONTINUE
        CLOSE(UNIT=31)
```

```
      STOP
      END
C     TITLE:MINV;SUBROUTINE TO INVERT A REAL SQUARE MATRIX
      SUBROUTINE MINV(A,MDIM,N)
      DIMENSION A(MDIM,N)
      INTEGER L(100),M(100)
C
C     CONVERTED FROM SSP ROUTNE MINV
C
      DO 80 K=1,N
      L(K)=K
      M(K)=K
      BIGA=A(K,K)
      DO 20 J=K,N
      DO 20 I=K,N
10    IF(ABS(BIGA)-ABS(A(I,J))) 15,20,20
15    BIGA=A(I,J)
      L(K)=I
      M(K)=J
20    CONTINUE
      J=L(K)
      IF(J-K) 35,35,25
25    DO 30 I=1,N
      HOLD=-A(K,I)
      A(K,I)=A(J,I)
30    A(J,I)=HOLD
35    I=M(K)
      IF(I-K) 45,45,38
38    DO 40 J=1,N
      HOLD=-A(J,K)
      A(J,K)=A(J,I)
40    A(J,I)=HOLD
45    DO 55 I=1,N
      IF(I-K) 50,55,50
50    A(I,K)=A(I,K)/(-BIGA)
55    CONTINUE
      DO 65 I=1,N
      HOLD=A(I,K)
      DO 65 J=1,N
      IF(I-K) 60,65,60
60    IF(J-K) 62,65,62
62    A(I,J)=HOLD*A(K,J)+A(I,J)
65    CONTINUE
      DO 75 J=1,N
      IF(J-K) 70,75,70
70    A(K,J)=A(K,J)/BIGA
75    CONTINUE
      A(K,K)=(1.0000000)/BIGA
80    CONTINUE
      K=N
100   K=K-1
```

```
        IF(K) 150,150,105
105     I=L(K)
        IF(I-K) 120,120,108
108     DO 110 J=1,N
        HOLD=A(J,K)
        A(J,K)=-A(J,I)
110     A(J,I)=HOLD
120     J=M(K)
        IF(J-K) 100,100,125
125     DO 130 I=1,N
        HOLD=A(K,I)
        A(K,I)=-A(J,I)
130     A(J,I)=HOLD
        GO TO 100
150     RETURN
        END
******************************************************
C       TITLE:ZBUS.F4;ACCEPTS DATA FROM TEST.DAT AND DSK.XFR
FROM
C               LOAD FLOW PROGRAM LDFLO.F4 AND FORMS ZBUS BY
C               ALGORITHM THIS WILL BE USED TO COMPUTE ALL
FAULT
C               CURRENTS.LOAD FLOW COMPUTED MAXIMUM LOAD
CURRENTS.
*********************************************************
*********************************************************
C       READ DATA FROM TEST.DAT
*********************************************************
        DIMENSION
ZD(300,3),ZM(300,2),RID(30,30,3),ZG(30),ZN(30),

1ID(200),RI(30),ZB(30,30),ZBO(30,30),ZMU(5,5),IDD(5),
ZL(30),
        2ZBR(30),ZGO(30),ZLO(30),EB(30)
        COMPLEX ZD,ZM,ZG,T1,ZN,ZN1,ZNI,ZNJ,ZG1,ZO,
        1ZB,ZBO,ZMU,ZBB,ZL,ZLL,ZBR,ZGO,ZLO,ZLLO,EB
        NNB=1
        EQUIVALENCE(ZBR,ZLO,EB)
        OPEN(UNIT=31,FILE='TEST.DAT')
C       OPEN(UNIT=32,FILE='DSK.XFR')
        READ(31,10)IB,EB1
10      FORMAT(I,F)
20      READ(31,30)NB,I,J,JJ,T1,ZG1,ZNI,ZNJ
30      FORMAT(4I,8F)
        IF(NB.EQ.0) GO TO 40
        DN=NB
        NN=NB
        ZD(NN,1)=CMPLX(DN,0.0)
        ZD(NN,2)=T1
        ZD(NN,3)=ZG1+3*ZN(I)
        ZN(I)=ZNI
```

```
        ZN(J)=ZNJ
        RID(I,J,JJ)=NN
        IF(NB.NE.0)GO TO 20
40      DO 45 I=1,IB
C       READ(32,43)T1
43      FORMAT(2F)
45      CONTINUE
        DO 48 I=1,IB
C       READ(32,46)J,T1
46      FORMAT(I,2F)
        EB(J)=T1
48      CONTINUE
49      READ(31,50)I,T1,XD1
50      FORMAT(I,3F)
        IF(I.EQ.0) GO TO 60
        IF(CABS(T1).EQ.0) GO TO 49
C       T1=CONJG(T1)
C       ZG(I)=3*(CABS(EB(I))**2)/T1
        IF(XD1.NE.0)ZG(I)=CMPLX(0.0,XD1)
        ZG0(I)=ZG(I)
        IF(I.NE.0) GO TO 49
******************************************************
C       ENTER THE ZERO SEQ IMP BUS   TO REF(GROUND)
*********************************************************
60      TYPE 62
62      FORMAT(' ENTER THE SYNC  ZERO SEQUENCE IMPEDENCE FOR
THE LOADS
        1AT EACH'/' BUS,3*ZN,MUST BE ADDED,IMPED IN PU..,')
        DO 65 I=1,IB
        ACCEPT 64,II,T1
        IF(II.EQ.0)GO TO 66
        ZG0(II)=T1
64      FORMAT(I,2F)
65      CONTINUE
66      DO 68 I=1,IB
        IF(CABS(ZG0(I)).EQ.0)ZG0(I)=ZG(I)
68      CONTINUE
*********************************************************
C       ENTER THE MUTUALLY COUPLED ELEMENTS
*********************************************************
        CLOSE(UNIT=31)
        TYPE 70
70      FORMAT(' ENTER THE MUTUALLY COUPLED ELEMENTS IN PER
UNIT'/'
        1 IN THE FOLLOWING FORMAT,TEST.DAT HAS THE ELEMENT
 S..,'/'
        2 (ELEMENT )SP TO(ELEMENT )SP(0 SEQ VALUE) ')
75      ACCEPT 80,EL,ELL,Z0
80      FORMAT(4F)
        IF(EL.EQ.0.0) GO TO 100
        II=II+1
```

```
        ZM(II,1)=CMPLX(EL,ELL)
        ZM(II,2)=Z0
        GO TO 75
*******************************************************
C       ALL ELEMENT DATA PREPARED PERFORM POSITIVE AND
NEGATIVE
C       SEQUENCE COMPUTATIONS TO FORM ZBUS
*********************************************************
*********************************************************
100     DO 500 IEL=1,NN
                BRA=0
                ID(IEL)=1
                DO 120 I=1,30
                        DO 120 J=1,30
                                DO 120 JJ=1,3
                                IF(RID(I,J,JJ).NE.IEL) GO TO
120
                                        IF(RI(I).EQ.0)GO TO
110
                                        IF(RI(J).EQ.0)GO TO
115
                                        BRA=1.0
                                        IPP=MIN0(I,J)
                                        IQQ=MAX0(I,J)
                                        GO TO 130
110                                     IF(RI(J).EQ.0)GO TO
115
                                        RI(I)=1.0
                                        IPP=J
                                        IQQ=I
                                        GO TO 130
115                                     RI(J)=1.0
                                        RI(I)=1.0
                                        IPP=MIN0(I,J)
                                        IQQ=MAX0(I,J)
                                        GO TO 130
120             CONTINUE
******************************************************
C       BRANCH OR LINK HAS BEEN DETERMINED BRA=0:BRANCH
**********************************************************
130             DO 195 IL=1,IB
                        ZBB=CMPLX(0.0,0.0)
181                     IF(BRA.NE.0) GO TO 186
                        IF(IQQ.EQ.IL) GO TO 190
                        ZBR(IL)=ZB(IPP,IL)
                        GO TO 190
186                     ZL(IL)=ZB(IPP,IL)-ZB(IQQ,IL)
190                     CONTINUE
195             CONTINUE
                ZBB=CMPLX(0.0,0.0)
                IF(BRA.NE.0) GO TO 250
```

```
                DO 200 IL=1,IB
                        ZB(IQQ,IL)=ZBR(IL)
                        ZB(IL,IQQ)=ZBR(IL)
200             CONTINUE
240             ZB(IQQ,IQQ)=ZB(IPP,IQQ)+ZD(IEL,2)
                ZBB=CMPLX(0.0,0.0)
                GO TO 300
250             ZLL=ZL(IPP)-ZL(IQQ)+ZD(IEL,2)
                ZBB=CMPLX(0.0,0.0)
                DO 290 IP=1,IB
                DO 290 IQ=1,IB
                                 IF(IP.GT.IQ)GO TO 290

ZB(IP,IQ)=ZB(IP,IQ)-(ZL(IP)*Z
L(IQ)
        1/ZLL)
                                 IF(IP.EQ.IQ) GO TO 290
                                 ZB(IQ,IP)=ZB(IP,IQ)
290     CONTINUE
300             DO 350 I=1,5
                DO 350 J=1,5
                        ZMU(I,J)=CMPLX(0.0,0.0)
                        IDD(I)=0
350             CONTINUE
                DO 400 I=1,IB
                DO 400 J=1,IB
                        PRINT 380,I,J,ZB(I,J)
380                     FORMAT(2I,2F)
400             CONTINUE
                NNB=1
500     CONTINUE
*****************************************************
C               ZBUS POS/NEG IS FORMED
*********************************************************
        DO 505 I=1,NN
                ID(I)=0
505     CONTINUE
        DO 510 I=1,IB
                RI(I)=0.0
510     CONTINUE
        DO 1000 IEL=1,NN
                BRA=0
                ID(IEL)=1
                NNB=1
                DO 620 I=1,30
                        DO 620 J=1,30
                                 DO 620 JJ=1,3
                                 IF(RID(I,J,JJ).NE.IEL) GO TO
620
                                         IF(RI(I).EQ.0)GO TO
610
```

```
                                        IF(RI(J).EQ.0) GO TO
615
                                        BRA=1.0
                                        IPP=MIN0(I,J)
                                        IQQ=MAX0(I,J)
                                        GO TO 625
610                                     IF(RI(J).EQ.0) GO TO
615
                                        RI(I)=1.0
                                        IPP=J
                                        IQQ=I
                                        GO TO 625
615                                     RI(I)=1.0
                                        RI(J)=1.0
                                        IPP=I
                                        IQQ=J
                                        GO TO 625
620             CONTINUE
*********************************************************
C       SEARCH FOR MUTUAL COUPLING IN ZERO SEQUENCE
*********************************************************
625             IDD(1)=IEL
                DO 650 I=1,NN
                        DM=IEL
                        IF(REAL(ZM(I,1)).EQ.DM)MUT=1
                        IF(AIMAG(ZM(I,1)).EQ.DM)MUT=1
                        IF(MUT.NE.1)GO TO 650
                        MUT=0
                        II1=IFIX(REAL(ZM(I,1)))
                        II2=IFIX(AIMAG(ZM(I,1)))
                        IF(ID(II1).EQ.0)GO TO 650
                        IF(ID(II2).EQ.0)GO TO 650
                        IF(II1.EQ.IEL)GO TO 630
                        II3=II1
                        II1=II2
                        II2=II3
630                     IDD(1)=II1
                        NNB=NNB+1
                        IDD(NNB)=II2
                        DM=II2
                        DO 645 J1=1,NN

IF(REAL(ZM(J1,1)).EQ.DM)MUT=1

IF(AIMAG(ZM(J1,1)).EQ.DM)MUT=
1
                                IF(MUT.NE.1) GO TO 645
                                MUT=0
                                III1=IFIX(REAL(ZM(J1,1)))
                                III2=IFIX(AIMAG(ZM(J1,1)))
                                IF(ID(III1).EQ.0)GO TO 645
```

```
                                IF(ID(III2).EQ.0)GO TO 645
                                IF(III1.EQ.DM) GO TO 643
                                III3=III1
                                III1=III2
                                III2=III1
643                             IF(III2.EQ.III) GO TO 645
                                NNB=NNB+1
                                IDD(NNB)=III2
645                     CONTINUE
650             CONTINUE
                DO 660 I=1,NNB
                        ZMU(I,I)=ZD(IDD(I),3)
                        DO 660  J=1,NNB
                                DD1=IDD(I)
                                DD2=IDD(J)
                                DO 660  KK=1,NN

IF(ZM(KK,1).EQ.CMPLX(
DD1,DD2))
        1MUT=1

IF(ZM(KK,1).EQ.CMPLX(DD2,DD1)
) MUT=1
                                IF(MUT.NE.1)GO TO 660
                                        MUT=0
                                        ZMU(I,J)=ZM(KK,2)
                                        ZMU(J,I)=ZM(KK,2)
660             CONTINUE
*********************************************************
C       ZMU IS FORMED
*******************************************************
                DO 665 I=1,NNB
                DO 665 J=1,NNB
                PRINT 663,I,J,ZMU(I,J)
663             FORMAT(2I,2F)
665             CONTINUE
*********************************************************
                CALL CMXINV(ZMU,5,NNB)
********************************************************
                DO 668 I=1,NNB
                PRINT 667,(ZMU(I,J),J=1,NNB)
667             FORMAT(10F)
668             CONTINUE
                DO 695 IL=1,IB
                IF(NNB.EQ.1) GO TO 681
                ZBB=CMPLX(0.0,0.0)
                        DO 680  J4=1,NNB
                                J=J4+1
                                I=IDD(J)
                                IF(I.EQ.0) GO TO 680
                                DO 670 I1=1,30
```

```
                                        DO 670 I2=1,30
                                                DO 670
I3=1,3

IF(RI
D(I1,I2,I3
        1).NE.I)GO TO 670

IP=I1

IQ=I2
        ZBB=(ZMU(1,J)/ZMU(1,1))*(ZBO(IP,IL)-ZBO(IQ,IL))+ZBB
670                             CONTINUE
680                     CONTINUE
681                     IF(BRA.NE.0) GO TO 686
                        IF(IQQ.EQ.IL) GO TO 690
                        ZBR(IL)=ZBO(IPP,IL)+ZBB
                        GO TO 690
686                     ZL(IL)=ZBB+ZBO(IPP,IL)-ZBO(IQQ,IL)
690                     CONTINUE
695             CONTINUE
                ZBB=CMPLX(0.0,0.0)
                IF(BRA.NE.0) GO TO 750
                DO 700 IL=1,IB
                        ZBO(IQQ,IL)=ZBR(IL)
                        ZBO(IL,IQQ)=ZBR(IL)
700             CONTINUE
                IF(NNB.EQ.1) GO TO 740
                DO 730 J=2,NNB
                        I=IDD(J)
                        DO 720  I1=1,30
                                DO 720 I2=1,30
                                        DO 720 I3=1,3
                        IF(RID(I1,I2,I3).NE.I) GO TO 720
                                                IP=I1
                                                IQ=I2
720                     CONTINUE

ZBB=(ZMU(1,J)*(ZBO(IP,IQQ)-ZBO(IQ,IQQ
))/
        1ZMU(1,1))+ZBB
730             CONTINUE
740             ZBO(IQQ,IQQ)=ZBO(IPP,IQQ)+ZBB+1.0/ZMU(1,1)
                ZBB=CMPLX(0.0,0.0)
                GO TO 800
750             IF(NNB.EQ.1) GO TO 781
                DO 780 J=2,NNB
                        I=IDD(J)
                        DO 770  I1=1,30
                                DO 770  I2=1,30
                                        DO 770  I3=1,3
```

```
                                                IF(RID(I1,
        1I2,I3).NE.I)GO TO 770
                                                IP=I1
                                                IQ=I2
770                     CONTINUE

ZBB=(ZMU(I,J)*(ZL(IP)-ZL(IQ))/ZMU(1,1
))+ZBB
780             CONTINUE
781             ZLL=ZL(IPP)-ZL(IQQ)+ZBB+(1.0/ZMU(1,1))
                ZBB=CMPLX(0.0,0.0)
                DO 790 IP=1,IB
                DO 790 IQ=1,IB
                                IF(IP.GT.IQ)GO TO 790

ZBO(IP,IQ)=ZBO(IP,IQ)-(ZL(IP)
*ZL(IQ)
        1/ZLL)
                                IF(IP.EQ.IQ) GO TO 790
                                ZBO(IQ,IP)=ZBO(IP,IQ)
790     CONTINUE
800             DO 850 I=1,5
                DO 850 J=1,5
                        ZMU(I,J)=CMPLX(0.0,0.0)
                        IDD(I)=0
850             CONTINUE
                DO 900 I=1,IB
                DO 900 J=1,IB
                        PRINT 880,I,J,ZBO(I,J)
880                     FORMAT(2I,2F)
900             CONTINUE
                NNB=1
1000    CONTINUE
************************************************************
******
C               ZBUS ZERO IS FORMED
*****************************************************
        OPEN(UNIT=33,FILE='ZBUS.FOR')
        DO 1010 I=1,IB
        DO 1010 J=1,IB
        WRITE(33,1005)I,J,ZBO(I,J),ZB(I,J),ZB(I,J)
1005    FORMAT(2I,6F)
        PRINT 1007
1007    FORMAT(3X,'BUS  TO BUS ',3X,'ZERO SEQ ',5X,'POS
SEQ')
        PRINT 1008,I,J,ZBO(I,J),ZB(I,J)
1008    FORMAT(2I6,4F)
1010    CONTINUE
        CLOSE(UNIT=32)
*****************************************************
C       NOW TO ADD ALL LOADS. ALL LOADS WILL BE LINKS WITH
```

```
NO
C     MUTUAL COUPLING.
************************************************************
      DO 1200 I=1,IB
            IF(CABS(ZG(I)).EQ.0)GO TO 1200
            DO 1100 IL=1,IB
                  ZL(IL)=-ZB(I,IL)
                  ZLO(IL)=-ZBO(I,IL)
1100        CONTINUE
            ZLL=-ZL(I)+ZG(I)
            ZLLO=-ZLO(I)+ZGO(I)
            DO 1150 J=1,IB
                  DO 1150 K=1,IB
                        IF(J.GT.K)GO TO 1150

ZB(J,K)=ZB(J,K)-ZL(J)*ZL(K)/Z
LL
                        ZB(K,J)=ZB(J,K)

ZBO(J,K)=ZBO(J,K)-ZLO(J)*ZLO(
K)/ZLLO
                        ZBO(K,J)=ZBO(J,K)
1150        CONTINUE
1200  CONTINUE
      OPEN(UNIT=34,FILE='ZBUS.DAT')
      DO 1300 I=1,IB
            DO 1300 J=1,IB
      WRITE(34,1250)I,J,ZB(I,J),ZBO(I,J)
1250  FORMAT(2I,4F)
1300  CONTINUE
      CLOSE(UNIT=34)
      CLOSE(UNIT=33)
      OPEN(UNIT=36,FILE='MUT.CO')
      DO 1400 I=1,NN
            EL1=REAL(ZM(I,1))
            IF(EL1.EQ.0.0)GO TO 1322
            EL2=AIMAG(ZM(I,1))
            IEL1=IFIX(EL1)
            IEL2=IFIX(EL2)
            DO 1320 I1=1,IB
            DO 1320 J1=1,IB
            DO 1320 K1=1,IB
            IF(RID(I1,J1,K1).EQ.EL1)GO TO 1310
            IF(RID(I1,J1,K1).EQ.EL2)GO TO 1315
            GO TO 1320
1310        IBII=I1
            IBIJ=J1
            GO TO 1320
1315        IBJI=I1
            IBJJ=J1
1320        CONTINUE
```

```
1322    IF(EL1.EQ.0)GO TO 1400

WRITE(36,1350)EL1,EL2,IBII,IBIJ,IBJI,IBJJ,ZM(
I,2)
1350            FORMAT(2F,4I3,2F)
1400    CONTINUE
        WRITE(36,1450)
1450    FORMAT('  0.0 0.0 0 0 0 0 0.0 0.0')
        CLOSE(UNIT=36)
        STOP
        END
C       TITLE:CMXINV;SUBROUTINE FOR INVERT COMPLEX MATRIX
        SUBROUTINE CMXINV(A,MDIM,N)
        COMPLEX A(MDIM,N),BIGA,HOLD
        INTEGER L(100),M(100)
C
C       CONVERTED FROM SSP ROUTNE MINV
C
        DO 80 K=1,N
        L(K)=K
        M(K)=K
        BIGA=A(K,K)
        DO 20 J=K,N
        DO 20 I=K,N
10      IF(CABS(BIGA)-CABS(A(I,J))) 15,20,20
15      BIGA=A(I,J)
        L(K)=I
        M(K)=J
20      CONTINUE
        J=L(K)
        IF(J-K) 35,35,25
25      DO 30 I=1,N
        HOLD=-A(K,I)
        A(K,I)=A(J,I)
30      A(J,I)=HOLD
35      I=M(K)
        IF(I-K) 45,45,38
38      DO 40 J=1,N
        HOLD=-A(J,K)
        A(J,K)=A(J,I)
40      A(J,I)=HOLD
45      DO 55 I=1,N
        IF(I-K) 50,55,50
50      A(I,K)=A(I,K)/(-BIGA)
55      CONTINUE
        DO 65 I=1,N
        HOLD=A(I,K)
        DO 65 J=1,N
        IF(I-K) 60,65,60
60      IF(J-K) 62,65,62
62      A(I,J)=HOLD*A(K,J)+A(I,J)
```

```
65      CONTINUE
        DO 75 J=1,N
        IF(J-K) 70,75,70
70      A(K,J)=A(K,J)/BIGA
75      CONTINUE
        A(K,K)=(1.0000000,0.0000000)/BIGA
80      CONTINUE
        K=N
100     K=K-1
        IF(K) 150,150,105
105     I=L(K)
        IF(I-K) 120,120,108
108     DO 110 J=1,N
        HOLD=A(J,K)
        A(J,K)=-A(J,I)
110     A(J,I)=HOLD
120     J=M(K)
        IF(J-K) 100,100,125
125     DO 130 I=1,N
        HOLD=A(K,I)
        A(K,I)=-A(J,I)
130     A(J,I)=HOLD
        GO TO 100
150     RETURN
        END
********************************************************
C       TITLE:FALT.F4;TAKES DATA FROM ZBUS.DAT AND COMPUTES
C                     ALL FAULT CURRENTS AT ALL BUSES
********************************************************
        DIMENSION
ZB(30,30),ZBO(30,30),FCUR(300,3,2),EFALT(30,3,2)
        1,EB(30),IDELE(300),EIB(30)
        COMPLEX
ZB,ZBO,FCUR,EFALT,EB,EB1,ZB1,ZBO1,ZR1,ZN1,ZNI,ZNJ,
        1ZR2,ZN2,ZF,IDELE,FCUR1,FCUR2,ALCUR
        OPEN(UNIT=34,FILE='TEST.DAT')
        OPEN(UNIT=35,FILE='ZBUS.DAT')
        READ(34,5) IB,E1
5       FORMAT(I,F)
        IBB=IB*IB
        DO 7 I=1,IB
        EB(I)=CMPLX(E1,0.0)
7       CONTINUE
9       READ(34,10),NI,I,J,K
10      FORMAT(4I)
        IF(NI.EQ.0)GO TO 15
        IF(K.EQ.2)I=-I
        IF(K.EQ.3)J=-J
        RI=I
        RJ=J
        IDELE(NI)=CMPLX(RI,RJ)
```

```
        IF(NI.NE.0)GO TO 9
15      DO 30 II=1,IBB
        READ(35,25)I,J,ZB1,ZB01
25      FORMAT(2I,4F)
        ZB(I,J)=ZB1
        ZB0(I,J)=ZB01
30      CONTINUE
        CLOSE(UNIT=35)
        OPEN(UNIT=37,FILE='MUT.CO')
        OPEN(UNIT=38,FILE='FALT.DAT')
        OPEN(UNIT=39,FILE='EFLT.DAT')
        OPEN(UNIT=40,FILE='RELA.DAT')
        OPEN(UNIT=41,FILE='FCUR.DAT')
        OPEN(UNIT=42,FILE='DSK.XFR')
        OPEN(UNIT=43,FILE='LCUR.DAT')
        READ(40,32) RMVAB,RKVBA,IRR
32      FORMAT(2F,I)
        RKVBA=RKVBA*1E3
        FIPU=RMVAB*1E6/(SQRT(3.0)*RKVBA)
33      READ(40,34)IRR
34      FORMAT(I)
        IF(IRR.NE.0)GO TO 33
36      READ(40,37)J,E
37      FORMAT(I,F)
        IF(J.EQ.0)GO TO 38
        EIB(J)=E
        GO TO 36
38      TYPE 35
35      FORMAT(' ENTER THE FAULT
IMPEDENCE-REAL(SP)IMAGINARY..,')
        ACCEPT 40,ZF
40      FORMAT(2F)
        DO 500 IFLT=1,IB
        IF(CABS(ZB(IFLT,IFLT)).EQ.0)GO TO 500
                DO 50 I=1,IB
                        IF(I.EQ.IFLT)GO TO 45

EFALT(I,2,1)=SQRT(3.0)*(EB(I)-((ZB(I,
IFLT)*EB
        1(IFLT))/(ZF+ZB(IFLT,IFLT))))

EFALT(I,1,2)=-(SQRT(3.0)*EB(IFLT)*ZB0
(I,IFLT)/
        1(ZB0(IFLT,IFLT)+3*ZF+(2*ZB(IFLT,IFLT))))

EFALT(I,2,2)=SQRT(3.0)*EB(I)-((SQRT(3
 .0)*EB

1(IFLT)*ZB(I,IFLT))/(ZB0(IFLT,IFLT)+(2*(ZB(IFLT,IFLT)
)+3*ZF)))
```

```
EFALT(I,3,2)=EFALT(I,2,2)-(SQRT(3.0)*
EB(I))
                          GO TO 50
45
EFALT(IFLT,2,1)=SQRT(3.0)*(ZF*EB(IFLT
)/(ZF+
        1ZB(IFLT,IFLT)))

EFALT(IFLT,1,2)=-SQRT(3.0)*EB(IFLT)*Z
BO(IFLT,
        1IFLT)/(ZBO(IFLT,IFLT)+(2*ZB(IFLT,IFLT))+3*ZF)

EFALT(IFLT,2,2)=SQRT(3.0)*EB(IFLT)*(Z
BO(IFLT,

1IFLT)+ZB(IFLT,IFLT)+3*ZF)/(ZBO(IFLT,IFLT)+(2*ZB(IFLT
,IFLT))+3*
        2ZF)

EFALT(IFLT,3,2)=-SQRT(3.0)*EB(IFLT)*Z
B(IFLT,IFL
        1T)/(ZBO(IFLT,IFLT)+(2*ZB(IFLT,IFLT))+3*ZF)
50                CONTINUE
                  REWIND 34
                  REWIND 37

WRITE(39,53)(IFLT,I,EFALT(I,2,1),EFALT(I,1,2)
,
        1EFALT(I,2,2),EFALT(I,3,2),I=1,IB)
53                FORMAT(2I3,8F)
                  READ(34,55)II,E
55                FORMAT(I,F)
                  DO 90 K=1,IBB
                  READ(34,60)NN,I,J,KK,ZR1,ZN1,ZNI,ZNJ
60                FORMAT(4I,8F)
                  IF(NN.EQ.0)GO TO 91
                  FCUR(NN,2,1)=(EFALT(I,2,1)-EFALT(J,2,1))/ZR1

FCUR(NN,1,2)=(EFALT(I,1,2)-EFALT(J,1,2))/(ZN1
+3*
        1(ZNI+ZNJ))
                  FCUR(NN,2,2)=(EFALT(I,2,2)-EFALT(J,2,2))/ZR1
                  FCUR(NN,3,2)=(EFALT(I,3,2)-EFALT(J,3,2))/ZR1
                          DO 85 I2=1,IBB

READ(37,65)RN1,RN2,IBII,IBIJ,
IBJI,IBJJ
        1,ZN2
65                                FORMAT(2F,4I,2F)
                                  NN1=IFIX(RN1)
                                  NN2=IFIX(RN2)
```

```
                                IF(NN1.EQ.NN)GO TO 70
                                IF(NN2.EQ.NN) GO TO 74
                                IF(NN1.EQ.0)GO TO 86
                                GO TO 85
70      FCUR(NN,1,2)=((EFALT(IBJI,1,2)-EFALT(IBJJ,1,2))/
        1(ZN2))+FCUR(NN,1,2)
                                GO TO 85
74      FCUR(NN,1,2)=((EFALT(IBII,1,2)-EFALT(IBIJ,1,2))/
        1(ZN2))+FCUR(NN,1,2)
85                      CONTINUE
86                      REWIND 37
90              CONTINUE
91              REWIND 34
                DO 300 NN=1,IBB
                        IF(CABS(FCUR(NN,2,1)).EQ.0)GO TO 300

WRITE(38,250)IFLT,NN,FCUR(NN,2,1),FCU
R(NN,1,2)
        1,FCUR(NN,2,2),FCUR(NN,3,2)
250                     FORMAT(2I3,8F)
300             CONTINUE
                DO 400 NN=1,IBB
                        IF(CABS(FCUR(NN,2,1)).EQ.0)GO TO 400

FCUR1=(RKVBA/EIB(IFIX(ABS(REAL(IDELE(
NN)
        1)))))*FCUR(NN,2,1)

FCUR2=(RKVBA/EIB(IFIX(ABS(REAL(IDELE(
NN)
        1)))))*(FCUR(NN,1,2)+FCUR(NN,2,2)+FCUR(NN,3,2))
                        FCUR1=FCUR1*FIPU
                        FCUR2=FCUR2*FIPU
                        IRI=(IFIX(REAL(IDELE(NN))))
                        IRJ=(IFIX(AIMAG(IDELE(NN))))
                        CUR1=CABS(FCUR1)
                        CUR2=CABS(FCUR2)
                        IF(CUR1.LT.0.001) GO TO 400
                        WRITE(41,350)IFLT,NN,CUR1,CUR2
350                     FORMAT(2I,2F)
400             CONTINUE
500     CONTINUE
        DO 600 I=1,IBB
        READ(42,525)IL,JL,ALCUR
        IF(IL.EQ.0)GO TO 610
        IF(REAL(ALCUR).LE.0.0)GO TO 600
        BLCUR=CABS(ALCUR)
        CLCUR=FIPU*BLCUR*RKVBA/EIB(JL)
        BLCUR=FIPU*BLCUR*RKVBA/EIB(IL)
        WRITE(43,525)IL,JL,BLCUR,CLCUR
525     FORMAT(2I,2F)
```

```
600     CONTINUE
610     WRITE(41,602)
        WRITE(43,602)
602     FORMAT(' 0 0 0 0 0 ')
        CLOSE(UNIT=36)
        CLOSE(UNIT=37)
        CLOSE(UNIT=38)
        CLOSE(UNIT=39)
        CLOSE(UNIT=42)
        CLOSE(UNIT=43)
        STOP
        END
```

```
*****************************************************
C       TITLE:COORD.F4;USES DATA FROM INPUT.F4,FALT.F4 TO
COORDINATE
C       THE PROTECTION DEVICES FOR THE WHOLE DISTRIBUTION
SYSTEM.
****************************************************
        OPEN(UNIT=30,FILE='RELA.DAT')
        OPEN(UNIT=31,FILE='LOCUR.DAT')
        OPEN(UNIT=32,FILE='LCUR.DAT')
        OPEN(UNIT=33,FILE='FCUR.DAT')
        OPEN(UNIT=34,FILE='TEST.DAT')
        OPEN(UNIT=35,FILE='XFOR.DAT')
        OPEN(UNIT=36,FILE='DEV.DAT')
        OPEN(UNIT=37,FILE='ADEV.DAT')
        DIMENSION EB(30),CUREL(200),CURLO(30),Z(100)
        1,AIDEVL(30,4),AIDEV(100,7),XFR(200,2),IID(3),RID(100
)
        2,DEV(20,0/4)
        COMPLEX Z,AIDEV,C1,C2,Z1,Z2,RID,CUREL,CU1,CURLO,XFR
**************************************************
C       READ IN LOAD AND BUS VOLTAGE DATA
****************************************************
        READ(30,5)R
5       FORMAT(F)
        READ(34,10)IB,E1
10      FORMAT(I,F)
11      READ(30,12)I
12      FORMAT(I)
        IF(I.NE.0)GO TO 11
13      READ(30,14)I,E1
14      FORMAT(I,F)
        IF(I.EQ.0)GO TO 15
        EB(I)=E1
        GO TO 13
15      DO 20 I=1,IB
        READ(31,16)J,CU2
16      FORMAT(I,F)
        IF(J.EQ.0)GO TO 21
        CURLO(J)=CMPLX(CU2*1.25,0.0)
20      CONTINUE
21      DO 24 I=1,20
        READ(36,22)K,(DEV(I,J),J=0,4)
        IF(K.EQ.0)GO TO 23
22      FORMAT(I,5F)
24      CONTINUE
**************************************************
C       NOW TO SET ALL LOAD DEVICES
****************************************************
23      TYPE 25
```

```
25      FORMAT(' ALL LOAD CURRENTS ARE READY,USE DEVICE
NUMBER LIST,')
        DO 100 I=1,IB
        IF(CABS(CURLO(I)).EQ.0)GO TO 100
        ID=I
30      TYPE 35,ID
35      FORMAT(' SELECT THE DEVICE FOR THE LOAD AT
BUS',I3,'..,'$)
        ACCEPT 40,IDEV
        AIDEVL(ID,1)=FLOAT(IDEV)
40      FORMAT(I)
        CURLI=REAL(CURLO(I))
        CALL
SELECT(CURLI,EB(I),AIDEVL(I,1),AIDEVL(I,2),AIDEVL(I,3)
        1,AIDEVL(I,4))
100     CONTINUE
*************************************************
C       NOW TO BEGIN COORDINATING ELEMENTS IN THE
DISTRIBUTION
**************************************************
110     READ(34,120)NN,I,J,K,R,X
120     FORMAT(4I,2F)
        IF(NN.EQ.0)GO TO 130
        NE=NN
        CI=I
        CJ=J
        RID(NN)=CMPLX(CI,CJ)
        Z(NN)=CMPLX(R,X)
        GO TO 110
130     DO 140 IE=1,NE
        READ(32,135)I,J,CU1
135     FORMAT(2I,2F)
        IF(I.EQ.0)GO TO 140
        IA=0
        CALL SERCH(NE,RID,IE,I,J,IA,IID)
        Z1=0.0*Z1
        Z2=0.0*Z2
                DO 136 IEL=1,IA
                Z1=1/(Z2+Z(IID(IEL)))
                Z2=1/Z1
136             CONTINUE
                DO 138  IEL=1,IA
                CUREL(IID(IA))=(Z2)*(1/Z(IID(IEL)))*CU1
138             CONTINUE
                IF(IA.GT.1)TYPE 101,J
101             FORMAT(' CAUTION THERE IS MORE THAN ONE
SOURCE
        1OF POWER TO THE BUS'I4)
140     CONTINUE
```

```
        DO 150  I=1,NE
                READ(35,145)IJ,J,TINRSH,TWST1,TWST2,TS1
145             FORMAT(2I,4F)
                IF(IJ.EQ.0)GO TO 151
                CALL SERCH(NE,RID,I,IJ,J,IA,IID)
                IA=1
147                     XFR(IID(IA),1)=CMPLX(TS1,TINRSH)
                        XFR(IID(IA),2)=CMPLX(TWST1,TWST2)
        IA=0
150     CONTINUE
********************************************
C       NOW TO DETERMINE WHAT ELEMENTS HAVE LOAD BUSSES
CONNECTED
***************************************************
151     DO 190  I=1,NE
                IR=IFIX(REAL(RID(I)))
                IM=IFIX(AIMAG(RID(I)))
                IF(AIDEVL(IR,1).NE.0) GO TO 155
                IF(AIDEVL(IM,1).NE.0)GO TO 156
                GO TO 190
155             II=IFIX(REAL(RID(I)))
                JJ=IFIX(AIMAG(RID(I)))
                GO TO 157
156             II=IFIX(AIMAG(RID(I)))
                JJ=IFIX(REAL(RID(I)))
157             CII=II
                CJJ=JJ
                AIDEV(I,1)=CMPLX(CJJ,CII)
                AIDEV(I,2)=CMPLX(0.0,AIDEVL(II,1))
                AIDEV(I,4)=CMPLX(AIDEVL(II,2),AIDEVL(II,3))
                IF(AIDEVL(II,1).GE.35)AIDEV(I,5)=CMPLX(0.0,
        1AIDEVL(II,4))
        IF(AIDEVL(II,1).LT.0)AIDEV(I,7)=CMPLX(0.0,AIDEVL(II,4
))
159             TYPE 158,JJ,II
158             FORMAT(' SELECT DEVICE FOR
BUS',I3,'PROTECTING ELEMENT
        1FEEDING'//' LOAD AT BUS',I4,'..,'$)
                ACCEPT 160,IDEV
160     FORMAT(I)
                AIDEV1=FLOAT(IDEV)
                DURR=AIMAG(AIDEV(I,4))
                CURR=REAL(AIDEV(I,4))
                IF(AIMAG(AIDEV(I,2)).GT.0)CURR=10**(DURR)
                CURR=REAL(CUREL(I))
        CALL SELECT(CURR,EB(JJ),AIDEV1,AIDEV2,AIDEV3,CT1)
                CALL
SETDEV(CUREL,I,JJ,II,AIDEV1,AIDEV2,AIDEV3,
        1AIDEV,XFR(I,1),XFR(I,2),RE,EB(JJ),EB(II),RID,CT1,DEV
```

```
)
              CT1=0.0
              IF(RE.NE.0)GO TO 159
190    CONTINUE
       IEL=0
200    NEL=NE-IEL
       IF(REAL(AIDEV(NEL,1)).NE.0)GO TO 300
       INDEX=1
       I=IFIX(REAL(RID(NEL)))
       J=IFIX(AIMAG(RID(NEL)))
       IF(AIMAG(AIDEV(NEL,2)).NE.0)GO TO 250
205    TYPE 210,J,NEL
       RE=0
210    FORMAT(' ENTER THE DEVICE   FOR THE BUS',I3,'OF
ELEMENT',I4)
       ACCEPT 230,IDEV
230    FORMAT(I)
       IF(IDEV.EQ.100)GO TO 400
       AIDEV1=FLOAT(IDEV)
       CURR=AIMAG(CUREL(NEL))
       CALL SELECT(CURR,EB(J),AIDEV1,AIDEV2,AIDEV3,CT1)
       IF(AIDEV1.GT.34)AIDEV(NEL,5)=CMPLX(0.0,.414973)
       CALL SETDEV(CUREL,NEL,I,J,AIDEV1,AIDEV2,AIDEV3,
       1AIDEV,XFR(NEL,1),XFR(NEL,2),RE,EB(I),EB(J),RID,CT1,D
EV)
       IF(RE.NE.0)GO TO 205
250    TYPE 210,I,NEL
       ACCEPT 230,IDEV
       AIDEV1=FLOAT(IDEV)
       RE=0
       CURR=REAL(CUREL(NEL))
       CALL SELECT(CURR,EB(I),AIDEV1,AIDEV2,AIDEV3,CT1)
       IF(AIDEV1.GT.34)AIDEV(NEL,5)=AIDEV(NEL,5)+CMPLX(.4149
73,0.0)
       CALL SETDEV(CUREL,NEL,I,J,AIDEV1,AIDEV2,AIDEV3,
       1AIDEV,XFR(NEL,1),XFR(NEL,2),RE,EB(I),EB(J),RID,CT1,D
EV)
       IF(RE.NE.0)GO TO 250
300    IEL=IEL+1
       IF(IEL.LT.NE) GO TO 200
       IF(INDEX.EQ.0)GO TO 400
       IEL=0
       INDEX=0
       GO TO 200
400    TYPE 410
       PRINT 410
410    FORMAT(' DEVICE LISTING')
       DO 440 I=1,NE
       TYPE 420,I,(AIDEV(I,J),J=1,7)
```

```
        PRINT 420,I,(AIDEV(I,J),J=1,7)
        WRITE(37,420)I,(AIDEV(I,J),J=1,7)
420     FORMAT(I,14F)
440     CONTINUE
        WRITE(37,442)
442     FORMAT(' 0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0
        1 0.0 0.0 0.0')
2000    CLOSE(UNIT=30)
        CLOSE(UNIT=31)
        CLOSE(UNIT=32)
        CLOSE(UNIT=33)
        CLOSE(UNIT=34)
        CLOSE(UNIT=36)
        CLOSE(UNIT=37)
        STOP
        END
*************************************************
*************************************************
C       SUBROUTINE SERCH;FINDS THE ELEMENT CORRESPONDING TO
TWO
C               BUSSES.
*************************************************
        SUBROUTINE SERCH(NE,RID,NN,I,J,IA,IID)
        DIMENSION RID(100),IID(3)
        COMPLEX RID,C1,C2
        IA=0
        DO 30   IEL=1,100
                CJ=J
                CI=I
                C1=CMPLX(CI,CJ)
                C2=CMPLX(CJ,CI)
                IF(RID(IEL).EQ.C1)GO TO 20
                IF(RID(IEL).EQ.C2)GO TO 20
                GO TO 30
20      IA=IA+1
        IID(IA)=IEL
30      CONTINUE
31      RETURN
        END
*************************************************
***********************************************
C       SUBROUTINE REFALT;READS FCUR.DAT TO OUTPUT FAULT
CURRENTS
C               AT THE BUSSES
************************************************
        SUBROUTINE REFALT(IFALT,NEL,FALT,CUREL)
10      READ(33,15)IEF,NN,TFAL,SFAL
15      FORMAT(2I,2F)
```

```
        IF(IEF.EQ.0)GO TO 20
        IF(IEF.NE.IFALT)GO TO 10
        IF(NN.NE.NEL)GO TO 10
        FALT1=TFAL
        FALT2=SFAL
        FALT=AMIN1(FALT1,FALT2)
        FLOCUR=3.0*CUREL
        IF(FALT.LE.FLOCUR)FALT=AMAX1(FALT1,FALT2)
        FALT=AMAX1(FALT1,FALT2)
20      REWIND 33
        RETURN
        END
****************************************************
C       SUBROUTINE SETDEV; SETS THE DEVICE INSTANTANEOUS
ELEMENT
C               DEPENDING ON THE FAULT CURRENT AT THAT BUS
DOES
C               COORDINATE WITH OTHER DEVICES
***************************************************
        SUBROUTINE
SETDEV(CUREL,IEL,I,J,AIDEV1,AIDEV2,AIDEV3,AIDEV
        1,XFR1,XFR2,RE,EB1,EB2,RID,CT,DEV)
        DIMENSION
AIDEV(100,7),CUREL(100),DEV(20,0/4),RID(100)
        COMPLEX AIDEV,XFR1,XFR2,CUREL,RID
        IDEV1=IFIX(AIDEV1)
30      IF(IDEV1.LT.0)CALL
SETREL(CUREL,IEL,I,J,AIDEV,AIDEV1,
        1AIDEV2,AIDEV3,XFR1,XFR2,RE,EB1,EB2,RID,CT,DEV)
        IF(IDEV1.LT.0)GO TO 100
        IF(IDEV1.EQ.99)GO TO 100
        IF(IDEV1.LT.35.AND.IDEV1.GT.0)CALL SETFUS(CUREL(IEL
        1),I,J,IEL,AIDEV,IDEV1,AIDEV2,AIDEV3,XFR1,XFR2,RE,EB1
,EB2,
        2DEV,RID)
        IF(IDEV1.GE.35.AND.IDEV1.LT.99)CALL
SETCKB(CUREL(IEL),
        1I,J,IEL,AIDEV,IDEV1,AIDEV2,AIDEV3,CT,EB1,EB2)
100     RETURN
        END
**********************************************
*************************************************
C       SUBROUTINE SETFUS;COORDINATES AND SELECTS A FUSE
**************************************************
        SUBROUTINE
SETFUS(CUREL,I,J,IEL,AIDEV,IDEV1,AIDEV2,AIDEV3
        1,XFR1,XFR2,RE,EB1,EB2,DEV,RID)
        DIMENSION
AIDEV(100,6),CUREL(100),DEV(20,0/4),RID(100)
```

```
        COMPLEX AIDEV,XFR1,XFR2,CUREL,RID
        JJ=IFIX(AIMAG(AIDEV(IEL,2)))
        IF(JJ.EQ.0)CALL XFDEV(1,J,IEL,AIDEV,COOF)
        IF(JJ.NE.0)CALL XFDEV(2,J,IEL,AIDEV,COOF)
        IF(JJ.EQ.0)CURR=AIMAG(CUREL(IEL))
        IF(JJ.NE.0)CURR=REAL(CUREL(IEL))
        IF(JJ.EQ.0.AND.COOF.LT.CURR)COOF=CURR
        IF(JJ.NE.0)COOF=(EB2/EB1)*COOF
        IF(JJ.NE.0.AND.COOF.LT.CURR)COOF=CURR
        IF(IDEV1.LE.15) CALL SFUSE(COOF,IDEV1,OFSET,OFSET1)
        IF(IDEV1.GT.15) CALL SBORFU(COOF,IDEV1,OFSET,OFSET1)
        IF(JJ.EQ.0) GO TO 20
        CI=I
        CJ=J
        AIDEV(IEL,1)=CMPLX(CI,CJ)
        AIDEV(IEL,2)=AIDEV(IEL,2)+CMPLX(FLOAT(IDEV1),0.0)
        AIDEV(IEL,3)=CMPLX(OFSET,OFSET1)
        GO TO 25
20      CI=I
        CJ=J
        AIDEV(IEL,1)=CMPLX(CI,CJ)
        AIDEV(IEL,2)=CMPLX(0.0,FLOAT(IDEV1))
        AIDEV(IEL,4)=CMPLX(OFSET,OFSET1)
        GO TO 50
25      IF(CABS(XFR1).EQ.0)GO TO 50
        AX=ALOG10((AIMAG(XFR1))/3.0)
        IF(REAL(AIDEV(IEL,2)).LE.15)GO TO 35
        IF(OFSET.LE.AX)CUR=10**AX
        IF(OFSET.LE.AX)CALL SBORFU(CUR,IDEV,OFSET,OFSET1)
        GO TO 40
35      IF(OFSET.LE.AX)CUR=10**AX
        IF(OFSET.LE.AX)CALL SFUSE(CUR,IDEV,OFSET,OFSET1)
40      AIDEV(IEL,2)=CMPLX(FLOAT(IDEV),AIMAG(AIDEV(IEL,2)))
        AIDEV(IEL,3)=CMPLX(OFSET,OFSET1)
        AR=ALOG10(REAL(XFR2))
        CALL FUSE(2,REAL(AIDEV(IEL,2)),AR,OFSET1,TVAL1)
        TVAL=10**TVAL1
        IF(TVAL.GT.2.0)GO TO 43
        GO TO 50
43      TYPE 45
45      FORMAT(' UNABLE TO PUT FUSE IN THIS LOCATION
SUBSTITUE
        1ANOTHER DEVICE')
        RE=1.0
        GO TO 300
50      CALL REFALT(J,IEL,FALT,CUREL(IEL))
        PAUSE 55
        IF(JJ.EQ.0)CALL
SFALT1(AIDEV,CUREL,RID,IEL,I,J,TVAL1,DEV,
```

```
        1CVAL)
        IF(JJ.NE.0)CALL SFALT2(AIDEV,IEL,I,J,TVAL1,DEV,
        1FALT,CVAL)
        PAUSE 60
        ADEV=FLOAT(IDEV1)
55      CALL FUSE(1,ADEV,CVAL,OFSET,TVAL)
        IF(TVAL.GT.TVAL1)GO TO 200
        ADEV=ADEV+1
        NEGDEV=-IFIX(ADEV)
        IF(ADEV.LE.15)CALL SFUSE(COOF,NEGDEV,OFSET,OFSET1)
        IF(ADEV.GT.15)CALL SBORFU(COOF,NEGDEV,OFSET,OFSET1)
        GO TO 55
200     IF(JJ.NE.0)GO TO 250
        AIDEV(IEL,2)=CMPLX(0.0,ADEV)
        AIDEV(IEL,4)=CMPLX(OFSET,OFSET1)
        GO TO 300
250     AIDEV(IEL,2)=CMPLX(ADEV,AIMAG(AIDEV(IEL,2)))
        AIDEV(IEL,3)=CMPLX(OFSET,OFSET1)
300     RETURN
        END
***********************************************
**********************************************
C       SUBROUTINE FUSE;COMPUTES TIME VALUE FOR A GIVEN LOAD
CURRENT
***********************************************
        SUBROUTINE FUSE(ID,ADEV,CVAL,OFSET,TVAL)
        TVAL=0.0
        IDEV=IFIX(ADEV)
        IF(IDEV.GT.15)GO TO 50
        X=CVAL-OFSET
        IF(ID.NE.1)GO TO 40
        TVAL=4.0480+(-36.4234*X)+(223.7526*(X**2))+(-751.279*
(X**3))
        1+(1347.01*(X**4))+(-1222.67*(X**5))+(440.813*(X**6))
        TVAL=TVAL-2.0
        GO TO 100
40      X=CVAL-OFSET+.1
        TVAL=6.426+(-37.922*X)+(121.909*(X**2))+(-213.420*(X*
*3))+
        1(201.771*(X**4))+(-96.9836*(X**5))+(18.6046*(X**6))
        TVAL=TVAL-2.0
        IF(X.GT.1.4)TVAL=-1.95
        GO TO 100
50      X=CVAL-OFSET
        IF(ID.NE.1)GO TO 70
        TVAL=3.87698+(-4.262768*X)+(1.219340*(X**2))
        TVAL=TVAL-2.0
        GO TO 100
70      X=CVAL-OFSET+.1
```

```
        TVAL=4.614840+(-6.183224*X)+(3.23181*(X**2))+(-.60571
36*(X**3))
        TVAL=TVAL-2.0
100     RETURN
        END
************************************************
*************************************************
C       SUBROUTINE RELAY;COMPUTES TIME VALUE FOR A GIVEN
CURRENT
C               VALUE FOR A SPECIFIC RELAY
*********************************************
        SUBROUTINE RELAY(NDEV,CURR,TVAL,TDSET,DEV,AMST)
        DIMENSION DEV(20,0/4)
        TVAL=0.0
        X=ALOG10(CURR)
        IF(X.GT.1.2)X=1.2
        DO 10 I=0,4
        TVAL=TVAL+DEV(NDEV,I)*(X**I)
10      CONTINUE
        IF(NDEV.EQ.7)TVAL=TVAL-2.0
        TVAL=TVAL+TDSET
        IF(CURR.GT.AMST.AND.AMST.NE.0)TVAL=-2.0
        RETURN
        END
************************************************
C       SUBROUTINE CKTBKR;COMPUTES TIME VALUE FOR A GIVEN
CURRENT
C               VALUE
***********************************************
        SUBROUTINE CKTBKR(ID,MDEV,DVAL,OFSET,TVAL,AMVAL)
        CVAL=DVAL-OFSET
        IF(ID.NE.1)GO TO 50
        CVAL=CVAL+.1122
        IF(AMVAL.EQ.0.0)AMVAL=10
        IF(CVAL.GT.AMVAL)GO TO 90
        TVAL=5.76908+(-10.88261*(CVAL))+(23.65419*(CVAL**2))+
(-34.7922
        1*(CVAL**3))+(33.77472*(CVAL**4))+(-21.14536*(CVAL**5
))+(6.4
        231778*(CVAL**6))
        TVAL=TVAL-2.0
        GO TO 100
50      CVAL=CVAL+.2315
        AMVAL=AMVAL+.075
        IF(CVAL.GT.AMVAL)GO TO 95
        TVAL=5.8596722+(-4.394335*CVAL)+(-.50690284*(CVAL**2)
)+(10.40
        12739*(CVAL**3))+(-17.064956*(CVAL**4))+(13.011543*(C
VAL**5))
```

```
        2+(-4.010724*(CVAL**6))
        TVAL=TVAL-2.0
        GO TO 100
90      TVAL=-2.0
        GO TO 100
95      TVAL=ALOG10(.025)
100     RETURN
        END
*******************************************
********************************************
C       SUBROUTINE SETREL;COORDINATES AND SELECTS A RELAY
*********************************************
        SUBROUTINE
SETREL(CUREL,IEL,I,J,AIDEV,AIDEV1,AIDEV2,AIDEV3,
        1XFR1,XFR2,RE,EB1,EB2,RID,CT,DEV)
        DIMENSION
AIDEV(100,7),IIDD(30),RID(100),IID(3),CUREL(100)
        1,DEV(20,0/4),FT(30,2)
        COMPLEX AIDEV,CUREL,XFR1,XFR2,RID
        COOF=0.0
        CURR=0.0
        CDUR=0.0
        TYPE 5
5       FORMAT(' DOES THIS RELAY HAVE AN INST. ELEMENT,IF
YES 0')
        ACCEPT 7,IL
7       FORMAT(I)
        JJ=IFIX(AIMAG(AIDEV(IEL,2)))
        MDEV=IFIX(-AIDEV1)
        IF(JJ.EQ.0)AIDEV(IEL,1)=CMPLX(FLOAT(I),FLOAT(J))
        IF(JJ.EQ.0)AIDEV(IEL,2)=CMPLX(0.0,AIDEV1)
        IF(JJ.NE.0)AIDEV(IEL,2)=CMPLX(AIDEV1,0.0)+AIDEV(IEL,2
)
        IF(JJ.EQ.0)CALL XFDEV(1,J,IEL,AIDEV,COOF)
        IF(JJ.NE.0)CALL XFDEV(2,J,IEL,AIDEV,COOF)
        IF(JJ.EQ.0)CURR=AIMAG(CUREL(IEL))
        IF(JJ.NE.0)CURR=REAL(CUREL(IEL))
        IF(JJ.EQ.0.AND.COOF.LT.CURR)COOF=CURR
        IF(JJ.NE.0)COOF=(EB2/EB1)*COOF
        CDUR=ALOG10(COOF)
        IF(JJ.NE.0.AND.COOF.LT.CURR)COOF=CURR
        IF(CDUR.GT.AIDEV2)AIDEV1=4.0
        IF(CDUR.GT.AIDEV2.AND.JJ.EQ.0)CALL
SELECT(COOF,EB2,AIDEV1,
        1AIDEV2,AIDEV3,CT)
        IF(CDUR.GT.AIDEV2.AND.JJ.NE.0)CALL
SELECT(COOF,EB1,AIDEV1,
        1AIDEV2,AIDEV3,CT)
        CURL=AIMAG(CUREL(IEL))
```

```
      IF(JJ.NE.0)AIDEV(IEL,3)=CMPLX(AIDEV2,AIDEV3)
      IF(JJ.EQ.0)AIDEV(IEL,4)=CMPLX(AIDEV2,AIDEV3)
      IF(JJ.NE.0)CURL=REAL(CUREL(IEL))
      IF(JJ.NE.0)AIDEV(IEL,7)=CMPLX(CT,0.0)+AIDEV(IEL,7)
      IF(JJ.EQ.0)AIDEV(IEL,7)=CMPLX(0.0,CT)
      IF(IL.NE.0)GO TO 100
      CALL REFALT(J,IEL,FALT1,CURL)
      IF(JJ.NE.0)GO TO 32
      FALT1=FALT1*(EB1/EB2)
      FALT1=FALT1/(10**(REAL(AIDEV(IEL,4))))
      IN=0
      DO 10 JI=1,30
            CALL SERCH(1,RID,IEL,J,JI,IA,IID)
            IF(IA.EQ.0)GO TO 10
            DO 8 JA=1,IA
                  IN=IN+1
                  IIDD(IN)=IID(IA)
8           CONTINUE
10    CONTINUE
      DO 30 INN=1,IN
            ID=IIDD(INN)
            IF(ID.EQ.IEL) GO TO 30
            IJ=IFIX(AIMAG(RID(ID)))
            CURJ=AIMAG(CUREL(ID))
            CALL REFALT(IJ,ID,FALT,CURJ)
            FALT2=ALOG10(FALT)
            AID=REAL(AIDEV(ID,2))
            IAID=IFIX(AID)
            IF(IAID.GT.0)GO TO 16
            IF(IAID.LT.0)FALTO=FALT/(10**(REAL(AIDEV(ID,3
))))
            NDEV=-IFIX(REAL(AIDEV(ID,2)))
            KFAL=IFIX(10*FALTO)
            DO 15 K=15,KFAL,10
                  RK=K*.1
                  IF(ALOG10(RK).GT.1.2)GO TO 16
                  CALL
RELAY(NDEV,RK,TVAL1,REAL(AIDEV(ID,6))
      1,DEV,REAL(AIDEV(ID,5)))
                  CALL
RELAY(MDEV,RK,TVAL2,TMSET,DEV,0.0)
                  TIMEO=(10**TVAL1)+.4
                  TIMER=TVAL2
                  TIMES=ALOG10(TIMEO)
                  TMSET1=TIMES-TIMER
                  IF(TMSET1.LE.0)GO TO 15
                  TMSET=TMSET1+TMSET
15          CONTINUE
16          IF(IAID.LT.0)GO TO 25
```

```
                OFSET=AIMAG(AIDEV(ID,3))
                FNSET=FALT2-OFSET
                IF(FNSET.GT.1.5)FALT2=OFSET+1.5
                KVAL=IFIX(10*OFSET)
                KFAL=IFIX(10*FALT2)
                DO 20 K=KVAL,KFAL
                        RK=K*.1
        IF(AID.LT.35)   CALL FUSE(2,AID,RK,OFSET,TVAL1)
        IF(AID.GE.35)CALL
CKTBKR(2,IFIX(AID),RK,OFSET,TVAL1,REAL(
        1AIDEV(IEL,5)))
                        CUR=(10**RK)/(10**(REAL(AIDEV(IEL,3))
))
                        CALL
RELAY(MDEV,CUR,TVAL2,TMSET,DEV,0.0)
                        TIMEO=10**TVAL1+.4
                        TIMER=TVAL2
                        TIMES=ALOG10(TIMEO)
                        TMSET1=TIMES-TIMER
                        IF(TMSET1.LE.0)GO TO 20
                        TMSET=TMSET+TMSET1
20              CONTINUE
25              GO TO 30
30      CONTINUE
        AIDEV(IEL,6)=CMPLX(0.0,TMSET)
        TMSET=0.0
        GO TO 45
32      AID=AIMAG(AIDEV(IEL,2))
        FALT2=ALOG10(FALT1)
        IF(AID.GT.0)GO TO 35
        FALT1=FALT1*(EB1/EB2)
        IF(AID.LT.0)FALT=FALT1/(10**(REAL(AIDEV(IEL,4))))
        NDEV=-IFIX(AIMAG(AIDEV(IEL,2)))
        KFAL=IFIX(FALT*10)
        DIF=KFAL-15
        ID=IFIX(DIF/50)
        IF(ID.LE.0)ID=1
        DO 33 K=15,KFAL,ID
                RK=K*.1
        RK2=RK*(EB2/EB1)*(10**(REAL(AIDEV(IEL,4)))/10**
        1(REAL(AIDEV(IEL,3))))
                IF(RK.GT.20)GO TO 34
                CALL
RELAY(NDEV,RK,TVAL1,AIMAG(AIDEV(IEL,6)),DEV,
        1AIMAG(AIDEV(IEL,5)))
                CALL RELAY(MDEV,RK2,TVAL2,TMSET,DEV,0.0)
                TIMEO=10**TVAL1+.4
                TIMES=ALOG10(TIMEO)
                TIMER=TVAL2
```

```
                  TMSET1=TIMES-TIMER
                  IF(TMSET1.LE.0)GO TO 33
                  TMSET=TMSET1+TMSET
33        CONTINUE
34        AIDEV(IEL,6)=AIDEV(IEL,6)+CMPLX(TMSET,0.0)
35        TMSET=0.0
          IF(AID.LT.0)GO TO 45
          OFSET=AIMAG(AIDEV(IEL,4))
          FNSET=FALT2-OFSET
          IF(FNSET.GT.1.5)FALT2=OFSET+1.5
          KVAL=IFIX(10*OFSET)
          KFAL=IFIX(10*FALT2)
          DO 40 K=KVAL,KFAL
                  RK=K*.1
          IF(AID.LT.35)CALL FUSE(2,AID,RK,OFSET,TVAL1)
          IF(AID.GE.35)CALL
CKTBKR(2,IFIX(AID),RK,OFSET,TVAL1,AIMAG(
          1AIDEV(IEL,5)))
                  CUR=RK-REAL(AIDEV(IEL,3))
                  CURTA=10**CUR
                  CUR=(EB2/EB1)*(10**RK)
                  CURTA=CUR/(10**(REAL(AIDEV(IEL,3))))
                  IF(CURTA.LT.1.5)GO TO 40
                  CALL RELAY(MDEV,CURTA,TVAL2,TMSET,DEV,0.0)
                  TIMEO=10**TVAL1+.4
                  TIMER=TVAL2
                  TIMES=ALOG10(TIMEO)
                  TMSET1=TIMES-TIMER
                  IF(TMSET1.LE.0)GO TO 40
                  TMSET=TMSET+TMSET1
40        CONTINUE
          AIDEV(IEL,6)=AIDEV(IEL,6)+CMPLX(TMSET,0.0)
45        TMSET=0.0
          IF(JJ.EQ.0.AND.FALT1.GT.6)FALT1=6
          IF(IL.EQ.0.AND.JJ.EQ.0)AIDEV(IEL,5)=CMPLX(0.0,FALT1)
          FALT1=FALT1/(10**REAL(AIDEV(IEL,3)))
          IF(FALT1.GT.6)FALT1=6
          IF(IL.EQ.0.AND.JJ.NE.0)AIDEV(IEL,5)=AIDEV(IEL,5)+CMPL
X(FALT1,
          10.0)
100       RETURN
          END
***********************************************
C         SUBROUTINE SFALT1;PROVIDES THE DEVICE WITH THE
LONGEST
C                 TIME AT FAULT,FIRST DEV IN ELEMENT.
**********************************************
          SUBROUTINE SFALT1(AIDEV,CUREL,RID,IEL,I,J,TVAL,DEV,
          1CVAL)
```

```
        DIMENSION AIDEV(100,6),CUREL(100),RID(100),IID(3),
        1IIDD(30),DEV(20,0/4)
        COMPLEX AIDEV,CUREL,RID
        DO 10 JI=1,30
                CALL SERCH(1,RID,IEL,J,JI,IA,IID)
                IF(IA.EQ.0)GO TO 10
                DO 8 JA=1,IA
                        IN=IN+1
                        IIDD(IN)=IID(IA)
8               CONTINUE
10      CONTINUE
        DO 30 INN=1,IN
                ID=IIDD(INN)
                IJ=IFIX(AIMAG(RID(ID)))
                CURJ=AIMAG(CUREL(ID))
                CALL REFALT(IJ,ID,FALT,CURJ)
                FALT2=ALOG10(FALT)
                AID=REAL(AIDEV(ID,2))
                IAID=IFIX(AID)
                IF(IAID.GT.32)GO TO 100
                FALT2=REAL(AIDEV(IEL,4))+.5
                FALT0=10**(FALT2)
                IF(IAID.LT.0)FALT=FALT0/(10**(REAL(AIDEV(ID,3
))))
                NDEV=-IFIX(REAL(AIDEV(ID,2)))
                IF(IAID.LT.0)CALL RELAY(NDEV,FALT,TVAL1,REAL
        1(AIDEV(ID,6)),DEV,REAL(AIDEV(ID,5)))
                FAL=REAL(AIDEV(IEL,4))+.5
                IF(FAL.LT.FALT2)FALT2=FAL
                IF(IAID.GT.0)CALL
FUSE(2,AID,FALT2,AIMAG(AIDEV
        1(ID,3)),TVAL1)
                IF(TVAL.GE.TVAL1)GO TO 25
                GO TO 30
25              IF(IAID.LT.0)CVAL=-FALT2
                CVAL=FAL
                TVAL=TVAL1
30      CONTINUE
100     RETURN
        END
**********************************************
**********************************************
C       SUBROUTINE SFALT2;AS IN 1,COMPUTES TIME FOR LOAD
SIDE OF
C               ELEMENT AT FALT.
********************************************
        SUBROUTINE SFALT2(AIDEV,IEL,I,J,TVAL1,DEV,FALT,CVAL)
        DIMENSION AIDEV(100,6),DEV(20,0/4)
        COMPLEX AIDEV
```

```
        FALT2=ALOG10(FALT)
        AID=AIMAG(AIDEV(IEL,2))
        FAL=REAL(AIDEV(IEL,3))+.5
        FALT=10**(FAL)
        FALT3=FALT/(10**(AIMAG(AIDEV(IEL,3))))
        IF(AID.LT.0)CALL
RELAY(NDEV,FALT3,TVAL1,AIMAG(AIDEV(IEL
        1,6)),DEV,AIMAG(AIDEV(IEL,5)))
        IF(FAL.LT.FALT2)FALT2=FAL
        IF(AID.GT.0)CALL FUSE(2,AID,FAL,AIMAG(AIDEV(IEL,4))
        1,TVAL1)
        CVAL=FALT2
        RETURN
        END
**************************************************
*************************************************
        SUBROUTINE XFDEV(NN,I,IEL,AIDEV,COOF)
        DIMENSION IID(30),AIDEV(100,6)
        COMPLEX AIDEV
        COOF=0.0
        IX=0
        IF(NN.NE.1)GO TO 50
        DO 30 IA=1,100
                AID=REAL(AIDEV(IA,1))
                IF(IFIX(AID).NE.I)GO TO 30
                IX=IX+1
                IID(IX)=IA
30      CONTINUE
        DO 32 K=1,IX
32      CONTINUE
        DO 40 JA=1,IX
                OFSET=AIMAG(AIDEV(IID(JA),3))
                IF(REAL(AIDEV(IID(JA),2)).LT.0)OFSET=REAL(AID
EV(
        1IID(JA),3))
                IF(REAL(AIDEV(IID(JA),2)).GE.35)OFSET=REAL(AI
DEV(
        1IID(JA),5))+OFSET-.05
                IF(AIMAG(AIDEV(IEL,2)).LT.0.AND.REAL(AIDEV(II
D
        1(JA),2)).GT.0.AND.REAL(AIDEV(IID(JA),2)).LT.35)
        2OFSET=OFSET+.15
                IF(COOF.GT.OFSET)GO TO 40
                COOF=OFSET
40      CONTINUE
        GO TO 60
50      IF(AIMAG(AIDEV(IEL,2)).GT.0)COOF=AIMAG(AIDEV(IEL,4))
        IF(AIMAG(AIDEV(IEL,2)).LT.0)COOF=REAL(AIDEV(IEL,4))
        IF(AIMAG(AIDEV(IEL,2)).GE.35)COOF=AIMAG(AIDEV(IEL,5))
```

```
-.075
        1+COOF
        IF(REAL(AIDEV(IEL,2)).LT.0.AND.AIMAG(AIDEV(IEL,2)).GT
.0
        1.AND.AIMAG(AIDEV(IEL,2)).LT.35)COOF=COOF+.15
60      COOF=COOF+.025
        DOOF=10**(COOF)
        COOF=DOOF
        RETURN
        END
*********************************************
************************************************
C       SUBROUTINE SETCKB;CORDINATES AND SELECTS A THERMAL
MAGNETIC
C               BREAKER.
***********************************************
**********************************************
        SUBROUTINE SETCKB(CUREL,I,JJ,IEL,AIDEV,IDEV,OFSET
        1,OFSET1,AMAG,EB1,EB2)
        DIMENSION AIDEV(100,7),CON(5)
        COMPLEX AIDEV,CUREL
        CON(1)=2.6
        CON(2)=3.5
        CON(3)=5.0
        CON(4)=6.0
        CON(5)=7.0
        J=IFIX(AIMAG(AIDEV(IEL,1)))
        IF(J.EQ.0)CALL XFDEV(1,J,IEL,AIDEV,COOF)
        IF(J.NE.0)CALL XFDEV(2,J,IEL,AIDEV,COOF)
        IF(J.EQ.0)DOOF=ALOG10(COOF)-AIMAG(AIDEV(IEL,5))
        IF(J.EQ.0)COOF=10**DOOF
        IF(J.NE.0)DOOF=ALOG10(COOF)-AIMAG(AIDEV(IEL,5))
        IF(J.NE.0)COOF=10**DOOF
        IF(J.EQ.0)CURR=AIMAG(CUREL)
        IF(J.NE.0)CURR=REAL(CUREL)
        IF(EB1.NE.EB2)GO TO 45
        IF(COOF.LT.CURR)COOF=CURR
        GO TO 46
45      RE2=(EB2/EB1)*COOF
        IF(RE2.GE.CURR)COOF=RE2
        IF(RE2.LT.CURR)COOF=CURR
46      CALL SBRKR(COOF,IDEV3,OFSET,OFSET1,DMMY)
        IF(IDEV.LT.IDEV3)IDEV=IDEV3
        IF(IDEV.GE.IDEV3)IDEV=IDEV+1
        CI=I
        CJ=JJ
        AIDEV(IEL,1)=CMPLX(CI,CJ)
        IF(J.EQ.0)AIDEV(IEL,2)=CMPLX(0.0,FLOAT(IDEV))
        IF(J.NE.0)AIDEV(IEL,2)=CMPLX(FLOAT(IDEV),0.0)+AIDEV
```

```
        1(IEL,2)
        IF(J.EQ.0)AIDEV(IEL,4)=CMPLX(OFSET,OFSET1)
        IF(J.NE.0)AIDEV(IEL,3)=CMPLX(OFSET,OFSET1)
        IF(J.EQ.0)FLT=4*AIMAG(CUREL)
        IF(J.NE.0)FLT=4*REAL(CUREL)
        SET=10**(OFSET-.12)
        DO 50 K=1,5
                TRP=CON(K)*SET
                IF(TRP.GT.FLT)GO TO 60
50      CONTINUE
60      IF(AMAG.NE.0)AMAG=ALOG10(CON(K))
        IF(J.EQ.0)AIDEV(IEL,5)=CMPLX(0.0,AMAG)
        IF(J.NE.0)AIDEV(IEL,5)=CMPLX(AMAG,0.0)+AIDEV(IEL,5)
        RETURN
        END
*********************************************
************************************************
C       SUBROUTINE SELECT;SELECTS A PARTICULAR DEVICE
**************************************************
        SUBROUTINE SELECT(CURLO,EB,ADEVL1,ADEVL2,ADEVL3,CT)
        IDEVL1=IFIX(ADEVL1)
54      IF(IDEVL1.EQ.0)GO TO 99
        IF(IDEVL1.EQ.1)GO TO 60
        IF(IDEVL1.EQ.2)GO TO 70
        IF(IDEVL1.EQ.4)GO TO 80
        IF(IDEVL1.GT.4)GO TO 53
        GO TO 58
53      TYPE 55
55      FORMAT(' NO SUCH DEVICE OR DEVICE ERROR, RE-ENTER
        1 DEVICE NR 1-4..,'$)
        ACCEPT 56, IDEVL1
56      FORMAT(I)
        GO TO 54
58      IF(EB.GT.600.0) GO TO 53
        CALL SBRKR(CURLO,IDEVL1,ADEVL2,ADEVL3,ADEVL4)
        ADEVL1=FLOAT(IDEVL1)
        CT=ADEVL4
        GO TO 100
60      CALL SFUSE(CURLO,IDEVL1,ADEVL2,ADEVL3)
        ADEVL1=FLOAT(IDEVL1)
        GO TO 100
70      IF(EB.LT.600)GO TO 53
        CALL SBORFU(CURLO,IDEVL1,ADEVL2,ADEVL3)
        ADEVL1=FLOAT(IDEVL1)
        GO TO 100
80      CALL SRELA(CURLO,IDEVL1,NTAP,NCT,PKUP,CT)
        ADEVL1=IDEVL1
        ADEVL2=PKUP
        ADEVL3=NCT*100+NTAP
```

```
        GO TO 100
99      ADEVL1=99
        ADEVL2=CURLO
        ADEVL3=CURLO
100     CONTINUE
        RETURN
        END
************************************************
C       SUBROUTINE SFUSE;USED TO SELECT A CURRENT LIMITING
FUSE
C               THIS MAY BE EITHER HIGH OR LOW VOLTAGE
*****************************************************
        SUBROUTINE SFUSE(CURLO,NDEV,OFSET,OFSET1)
        IF(NDEV.LT.0)GO TO 10
        DEV=-11.15221+(23.865*ALOG10(CURLO))-(16.487*(ALOG10(
CURLO
        1)**2))+(2.9288*(ALOG10(CURLO)**3))+(3.19672*(ALOG10(
CU
        2RLO)**4))-(1.48877*(ALOG10(CURLO)**5))+(.1776431*(AL
OG
        310(CURLO)**6))
        NDEV=IFIX(DEV)+1
        IF(NDEV.GT.15)NDEV=15
10      DEV=ABS(FLOAT(NDEV))
        OFSET=.9532-(.046433*DEV)+(.117885*(DEV**2))+(-.02935
2*(DEV
        1**3))+(.00337*(DEV**4))+(-.0001847*(DEV**5))+(.00000
39*(
        2DEV**6))
        OFSET1=OFSET+.1
        IF(NDEV.LT.-15)NDEV=-15
        IF(NDEV.GT.15)NDEV=15
        RETURN
        END
****************************************************
C       SUBROUTINE SBORFU;USED TO SELECT A BORIC ACID HIGH
VOLTAGE
C               FUSE
****************************************************
        SUBROUTINE SBORFU(CURLO,NDEV,OFSET,OFSET1)
        ALOGC=ALOG10(CURLO)
        IF(NDEV.LT.0)GO TO 10
        DEV=88.1643+(-215.2235*(ALOGC))+(188.225*(ALOGC**2))+
(-69.
        13463*(ALOGC**3))+(7.4695*(ALOGC**4))+(1.65257*(ALOGC
**5))+
        2(-.347786*(ALOGC**6))
        NDEV=IFIX(DEV)+1
        IF(NDEV.GT.19)NDEV=19.
```

```
10       DEV=ABS(FLOAT(NDEV))
         IF(NDEV.LT.0)DEV=DEV-15
         OFSET=1.366111+.1134579*DEV+(-.0012654*(DEV**2))+(.00
00213*
         1(DEV**3))
         OFSET1=OFSET+.1
         IF(NDEV.LT.-19)NDEV=-19
         IF(NDEV.GT.19)NDEV=19
         IF(NDEV.LT.0)NDEV=-NDEV+15
         IF(NDEV.GT.0)NDEV=NDEV+15
         RETURN
         END
**********************************************
C        SUBROUTINE SBRKR;USED TO SELECT A THERMAL MAGNETIC
BREAKER
**************************************************
         SUBROUTINE SBRKR(CURLO,NDEV,OFSET,OFSET1,AMAG)
         DIMENSION CON(5)
         CON(1)=2.6
         CON(2)=3.5
         CON(3)=4.6
         CON(4)=5.6
         CON(5)=7.0
         TYPE 10
10       FORMAT(' ENTER MANUFACTURER  (N.A. AT PRESENT USE
FOR FUTURE
         1EXPANSION)')
         TYPE 25
25       FORMAT(' INPUT TEMPERATURE FOR AMBIENT SHIFT,1 FOR
LESS THAN
         110 DEG C,2 FOR 60 DEG C,'//'<CR>FOR NO SHIFT..')
         ACCEPT 28,IOFF
28       FORMAT(I)
         IF(IOF.EQ.1)OFS=-.1
         IF(IOF.EQ.2)OFS=.05
         IF(MAN.NE.0)GO TO 80
         IF(CURLO.LE.600.)GO TO 30
         IF(CURLO.LE.800.)GO TO 50
         GO TO 70
30       IF(CURLO.LT.300.)MDEV=1
         IF(CURLO.LT.300.)OFSET=2.47712
         IF(CURLO.GT.300.0.AND.CURLO.LE.350.)MDEV=2
         IF(CURLO.GT.300.0.AND.CURLO.LE.350.)OFSET=2.54407
         IF(CURLO.GT.350.0.AND.CURLO.LE.400.)MDEV=3
         IF(CURLO.GT.350.0.AND.CURLO.LE.450.)OFSET=2.60206
         IF(CURLO.GT.400.0.AND.CURLO.LE.450.)MDEV=4
         IF(CURLO.GT.400.0.AND.CURLO.LE.450.)OFSET=2.65321
         IF(CURLO.GT.450.0.AND.CURLO.LE.500.)MDEV=5
         IF(CURLO.GT.450.0.AND.CURLO.LE.500.)OFSET=2.69897
```

```
        IF(CURLO.GT.500.)MDEV=6
        IF(CURLO.GT.500.)OFSET=2.77815
        GO TO 80
50      IF(CURLO.LE.700.)MDEV=7
        IF(CURLO.LE.700.)OFSET=2.8451
        IF(CURLO.GT.700.)MDEV=8
        IF(CURLO.GT.700.)OFSET=2.90309
        GO TO 80
70      IF(CURLO.LE.1000.)MDEV=9
        IF(CURLO.LE.1000.)OFSET=3.0
        IF(CURLO.GT.1000.)MDEV=10
        IF(CURLO.GT.1000.)OFSET=3.3
80      NDEV=MDEV+34
        OFSET=OFSET+OFS+.1122
        OFSET1=OFSET+.12
        TYPE 90
90      FORMAT(' DOES THE DEVICE HAVE A MAGNETIC
ELEMENT?(0)YES.')
        ACCEPT 95,NOEL
95      FORMAT(I)
        FCUR=4*CURLO
        SET=10**(OFSET-.12)
        DO 100 I=1,5
                TD=SET*CON(I)
                IF(TD.GT.FCUR)GO TO 105
100     CONTINUE
105     AMAG=ALOG10(CON(I))
        IF(NOEL.NE.0)AMAG=0.0
        RETURN
        END
**************************************************
C       SUBROUTINE SRELA;USED TO SELECT A RELAY
************************************************
        SUBROUTINE SRELA(CURLO,NDEV,NTAP,NCT,PKUP,CT)
        TYPE 10
10      FORMAT(' SELECT MANUFACTUER  (FUTURE EXPANSION)')
        TYPE 20
20      FORMAT(' SELECT THE WESTINGHOUSE CO RELAY  ,REFER TO
TABLE',$)
        ACCEPT 25,NUM
25      FORMAT(I)
        NDEV=-NUM
        IF(CURLO.LT.30)CT=10
        IF(CURLO.GE.30.AND.CURLO.LT.120)CT=30
        IF(CURLO.GE.120)CT=120
        IF(CT.EQ.10)NCT=1
        IF(CT.EQ.30)NCT=2
        IF(CT.EQ.120)NCT=3
        TAP=CURLO/CT
```

```
         IF(TAP.LE.1.0)GO TO 71
         IF(TAP.LE.1.2)GO TO 72
         IF(TAP.LE.1.5)GO TO 73
         IF(TAP.LE.2.0) GO TO 74
         IF(TAP.LE.2.5)GO TO 75
         IF(TAP.LE.3.0)GO TO 76
         IF(TAP.LE.3.5)GO TO 77
         IF(TAP.LE.4.0)GO TO 78
         IF(TAP.LE.5.0)GO TO 79
         IF(TAP.LE.6.0)GO TO 80
         IF(TAP.LE.7.0)GO TO 81
         IF(TAP.LE.8.0)GO TO 82
         IF(TAP.LE.10.0)GO TO 83
         GO TO 84
71       NTAP=1
         TAP=1.0
         GO TO 90
72       NTAP=2
         TAP=1.2
         GO TO 90
73       NTAP=3
         TAP=1.5
         GO TO 90
74       NTAP=4
         TAP=2.0
         GO TO 90
75       NTAP=5
         TAP=2.5
         GO TO 90
76       NTAP=6
         TAP=3.0
         GO TO 90
77       NTAP=7
         TAP=3.5
         GO TO 90
78       NTAP=8
         TAP=4.0
         GO TO 90
79       NTAP=9
         TAP=5.0
         GO TO 90
80       NTAP=10
         TAP=6.0
         GO TO 90
81       NTAP=11
         TAP=7.0
         GO TO 90
82       NTAP=12
         TAP=8.0
```

```
      GO TO 90
83    NTAP=13
      TAP=10.0
      GO TO 90
84    NTAP=14
      TAP=12.0
90    PKUP=ALOG10(TAP*CT)
      RETURN
      END
```

```
C       TITLE:PLOTD.F4;PLOTS DEVICES FROM COORD.F4
******************************************************
        OPEN(UNIT=30,FILE='ADEV.DAT')
        OPEN(UNIT=31,FILE='DEV.DAT')
        OPEN(UNIT=32,FILE='RELA.DAT')
        OPEN(UNIT=33,FILE='FCUR.DAT')
        OPEN(UNIT=34,FILE='TEST.DAT')
        EXTERNAL T25HAN
        DIMENSION EB(30),DEV(20,0/4),AIDEV(100,7),RID(100)
        COMPLEX AIDEV,RID
        READ(32,3)R
3       FORMAT(F)
5       READ(32,10)I
10      FORMAT(I)
        IF(I.NE.0)GO TO 5
15      READ(32,20)I,E1
20      FORMAT(I,F)
        IF(I.EQ.0)GO TO 25
        EB(I)=E1
        GO TO 15
25      DO 40 K=1,100
        READ(30,30)I,(AIDEV(I,J),J=1,7)
30      FORMAT(I,14F)
        IF(I.EQ.0)GO TO 45
40      CONTINUE
45      READ(34,50)IB,E1
50      FORMAT(I,F)
55      READ(34,60)NN,I,J,K,R,X
60      FORMAT(4I,2F)
        IF(NN.EQ.0) GO TO 70
        NE=NN
        CI=I
        CJ=J
        RID(NN)=CMPLX(CI,CJ)
        GO TO 55
70      DO 80 I=1,20
                READ(31,75)K,(DEV(I,J),J=0,4)
75      FORMAT(I,5F)
        IF(K.EQ.0)GO TO 400
80      CONTINUE
400     TYPE 410
410     FORMAT(' DEVICE LISTING')
        DO 440 I=1,NE
        TYPE 420,I,(AIDEV(I,J),J=1,7)
        PRINT 420,I,(AIDEV(I,J),J=1,7)
420     FORMAT(I,14F)
440     CONTINUE
445     TYPE 450
450     FORMAT(' DEVICES HAVE BEEN SET IF YOU WANT TO PLOT
```

```
GRAPH PAPER
        1 ENTER(0),IF NOT TYPE(1)')
        ACCEPT 456,IP
456     FORMAT(I)
        IF(IP.NE.0)GO TO 480
        TYPE 460
460     FORMAT(' THIS WILL PLOT CURVES BASED TO THE LOWEST
VOLTAGE
        1 IN THE SYSTEM.')
        TYPE 461
461     FORMAT(' INPUT THE CURRENT OFFSET FOR THE
GRAPH,INTEGER..')
        ACCEPT 462,IAOFF
462     FORMAT(I)
        CALL INITT
        CALL LOG(IAOFF)
        CALL FLUSH
480     TYPE 485
485     FORMAT(' INPUT ELEMENT  ..')
        ACCEPT 490,IEL
490     FORMAT(I)
        I=IFIX(REAL(RID(IEL)))
        J=IFIX(AIMAG(RID(IEL)))
        EB1=EB(1)
        DO 495 K=1,IB
                EB1=AMIN1(EB1,EB(K))
495     CONTINUE
550     FAC=ALOG10(EB(I)/EB1)-IAOFF
        FAC1=ALOG10(EB(J)/EB1)-IAOFF
        CUR=REAL(AIDEV(IEL,3))
        CUR1=REAL(AIDEV(IEL,4))
        CUR2=AIMAG(AIDEV(IEL,3))
        CUR3=AIMAG(AIDEV(IEL,4))
****************************************************
C       NOW TO PLOT LOAD SIDE OF THE ELEMENT
***************************************************
        CALL REFALT(J,IEL,FALT,900.0)
        IF(AIMAG(AIDEV(IEL,2)).GT.0) GO TO 585
        STL=REAL(AIDEV(IEL,4))
        STP=FALT/10**STL
        IF(AIMAG(AIDEV(IEL,5)).NE.0)STP=AIMAG(AIDEV(IEL,5))
        ST1=ALOG10(FALT)
        YM=(STL+FAC1)*190+175.0
        XM=0.0
        CALL MOVEA(XM,YM)
        STP=10*STP
        ITP=IFIX(STP)
        IF(ITP.LT.15)TYPE 575
575     FORMAT(' THE LOAD SIDE RELAY WILL NOT BE VERY
```

```
SENSITIVE
        1 TO A FAULT')
        IDEV=IFIX(ABS(AIMAG(AIDEV(IEL,2))))
        TYPE 570,IDEV,ITP
570     FORMAT(' IDEV,',2I)
        DIF=ITP-15.
        ID=IFIX(DIF/200.0)
        IF(ID.LE.0)ID=1
        TVAL3=10.0
        TDSET=AIMAG(AIDEV(IEL,6))
        DO 580 K=15,ITP,ID
                RK=.1*(FLOAT(K))
                CALL RELAY(IDEV,RK,TVAL,TDSET,DEV)
                IF(TVAL.GT.TVAL3)TVAL=TVAL3
                TVAL3=TVAL
                T=900-((TVAL+2.0)*180)
                R=(ALOG10(RK)+FAC1+STL)*190+175.0
                CALL DRAWA(T,R)
580     CONTINUE
        PAUSE 4
        IF(AIMAG(AIDEV(IEL,5)).NE.0)CALL DRAWA((900.0),R)
        GO TO 591
585     OFSET=CUR1
        M=IFIX(80*OFSET)
        ADEV=AIMAG(AIDEV(IEL,2))
        KIM=IFIX(100*(ALOG10(FALT)))
        DO 587 K=M,KIM
                CVAL=K*.01
        IF(ADEV.LT.35)CALL FUSE(1,ADEV,CVAL,OFSET,TVAL)
        IF(ADEV.GE.35)CALL
CKTBKR(1,IFIX(ADEV),CVAL,OFSET,TVAL,
        1AIMAG(AIDEV(IEL,5)))
                TVAL=TVAL+2.0
                IF(TVAL2.LT.TVAL.AND.ND.NE.0)GO TO 589
                TVAL2=TVAL
                ND=1
                CALL
DRAWA(900-((TVAL)*180),(CVAL+FAC1)*190+175.0)
587     CONTINUE
589     OFSET=CUR3
        CALL MOVEA(0.0,0.0)
        ND=0
        M=IFIX(80*OFSET)
        DO 588 K=M,KIM
                CVAL=K*.01
        IF(ADEV.LT.35)CALL FUSE(2,ADEV,CVAL,OFSET,TVAL)
        IF(ADEV.GE.35)CALL
CKTBKR(2,IFIX(ADEV),CVAL,OFSET,TVAL,
        1AIMAG(AIDEV(IEL,5)))
```

```
                TVAL=TVAL+2.0
                IF(TVAL2.LT.TVAL.AND.ND.NE.0)GO TO 591
                TVAL2=TVAL
                ND=1
                CALL
DRAWA(900-((TVAL)*180),(CVAL+FAC1)*190+175.0)
588     CONTINUE
****************************************************
C       NOW TO DRAW THE LINE SIDE
**************************************************
591     CALL MOVEA(0.0,0.0)
        AID=(REAL(AIDEV(IEL,2)))
        IF(AID.GT.0)GO TO 595
        IDEV=IFIX(-AID)
        STL=REAL(AIDEV(IEL,3))
        STP=FALT/10**STL
        IF(REAL(AIDEV(IEL,5)).NE.0)STP=REAL(AIDEV(IEL,5))
        YM=(STL+FAC)*190+175.0
        XM=0.0
        CALL MOVEA(XM,YM)
        STP=STP*10
        ITP=IFIX(STP)
        DIF=ITP-15
        ID=IFIX(DIF/200.0)
        IF(ID.LE.0)ID=1
        TVAL3=10.0
        TDSET=REAL(AIDEV(IEL,6))
        DO 590 K=15,ITP,ID
                RK=K*.1
                CALL RELAY(IDEV,RK,TVAL,TDSET,DEV)
                IF(TVAL.GT.TVAL3)TVAL=TVAL3
                TVAL3=TVAL
                CALL
DRAWA(900-((TVAL+2.0)*180),((ALOG10(RK)+STL
        1+FAC)*190+175.0))
590     CONTINUE
        IF(REAL(AIDEV(IEL,5)).NE.0)CALL
DRAWA((900.0),((ALOG10(RK)+
        1STL+FAC)*190+175))
595     IF(REAL(AIDEV(IEL,2)).LT.0)GO TO 598
        OFSET=CUR
        ADEV=REAL(AIDEV(IEL,2))
        ND=0
        M=IFIX(80*OFSET)
        KIM=IFIX(100*ALOG10(FALT))
        DO 597 K=M,KIM
                CVAL=K*.01
        IF(ADEV.LE.34)CALL FUSE(1,ADEV,CVAL,OFSET,TVAL)
        IF(ADEV.GE.35)CALL
```

```
CKTBKR(1,IFIX(ADEV),CVAL,OFSET,TVAL,
        1REAL(AIDEV(IEL,5)))
                TVAL=TVAL+2.0
                IF(TVAL2.LT.TVAL.AND.ND.NE.0)GO TO 601
                TVAL2=TVAL
                ND=1
                CALL
DRAWA(900-(TVAL*180),(CVAL+FAC)*190+175.0)
597     CONTINUE
601     OFSET=CUR2
        CALL MOVEA(0.0,0.0)
        ND=0
        M=IFIX(80*OFSET)
        DO 599 K=M,KIM
                CVAL=K*.01
        IF(ADEV.LE.34)CALL FUSE(2,ADEV,CVAL,OFSET,TVAL)
        IF(ADEV.GE.35)CALL
CKTBKR(2,IFIX(ADEV),CVAL,OFSET,TVAL,
        1REAL(AIDEV(IEL,5)))
                TVAL=TVAL+2.0
                IF(TVAL2.LT.TVAL.AND.ND.NE.0)GO TO 598
                TVAL2=TVAL
                ND=1
                CALL
DRAWA(900-(TVAL*180),(CVAL+FAC)*190+175.0)
599     CONTINUE
598     TYPE 600
600     FORMAT(' FINISHED??')
        ND=0
        CALL MOVEA(0.0,0.0)
        CALL FLUSH
        ACCEPT 610,I
610     FORMAT(I)
        IF(I.EQ.0)GO TO 2000
        GO TO 445
2000    CLOSE(UNIT=30)
        CLOSE(UNIT=31)
        CLOSE(UNIT=32)
        CLOSE(UNIT=33)
        CLOSE(UNIT=34)
        CALL FINITT(0,0)
        STOP
        END
**************************************************
C       SUBROUTINE REFALT;READS FCUR.DAT TO OUTPUT FAULT
CURRENTS
C               AT THE BUSSES
*************************************************
        SUBROUTINE REFALT(IFALT,NEL,FALT,CUREL)
```

```
10      READ(33,15)IEF,NN,TFAL,SFAL
15      FORMAT(2I,2F)
        IF(IEF.EQ.0)GO TO 20
        IF(IEF.NE.IFALT)GO TO 10
        IF(NN.NE.NEL)GO TO 10
        FALT1=TFAL
        FALT2=SFAL
        FALT=AMIN1(FALT1,FALT2)
        FLOCUR=3.0*CUREL
        IF(FALT.LE.FLOCUR)FALT=AMAX1(FALT1,FALT2)
        FALT=AMAX1(FALT1,FALT2)
20      REWIND 33
        RETURN
        END
****************************************************
C       SUBROUTINE FUSE;COMPUTES TIME VALUE FOR A GIVEN LOAD
CURRENT
******************************************************
        SUBROUTINE FUSE(ID,ADEV,CVAL,OFSET,TVAL)
        TVAL=0.0
        IDEV=IFIX(ADEV)
        IF(IDEV.GT.15)GO TO 50
        X=CVAL-OFSET
        IF(ID.NE.1)GO TO 40
        TVAL=4.0480+(-36.4234*X)+(223.7526*(X**2))+(-751.279*
(X**3))
        1+(1347.01*(X**4))+(-1222.67*(X**5))+(440.813*(X**6))
        TVAL=TVAL-2.0
        GO TO 100
40      X=CVAL-OFSET+.1
        TVAL=6.426+(-37.922*X)+(121.909*(X**2))+(-213.420*(X*
*3))+
        1(201.771*(X**4))+(-96.9836*(X**5))+(18.6046*(X**6))
        TVAL=TVAL-2.0
        IF(X.GT.1.4)TVAL=-1.95
        GO TO 100
50      X=CVAL-OFSET
        IF(ID.NE.1)GO TO 70
        TVAL=3.876979+(-4.262768*X)+(1.21940*(X**2))
        TVAL=TVAL-2.0
        GO TO 100
70      X=CVAL-OFSET+.1
        TVAL=4.614841+(-6.173224*X)+(3.231806*(X**2))+(-.6057
136*(X**3))
        TVAL=TVAL-2.0
100     RETURN
        END
************************************************
C       SUBROUTINE RELAY;COMPUTES TIME VALUE FOR A GIVEN
```

```
CURRENT
C               VALUE FOR A SPECIFIC RELAY
*********************************************
        SUBROUTINE RELAY(NDEV,CURR,TVAL,TDSET,DEV)
        DIMENSION DEV(20,0/4)
        TVAL=0.0
        X=ALOG10(CURR)
        IF(X.GT.1.2)X=1.2
        DO 10 I=0,4
        TVAL=TVAL+DEV(NDEV,I)*(X**I)
10      CONTINUE
        IF(NDEV.EQ.7)TVAL=TVAL-2.0
        TVAL=TVAL+TDSET
        RETURN
        END
*****************************************************
C       TITLE:LOG.LOG,MAKES LOG LOG FORM FOR GRAPH OF RELAY
CURVES
****************************************************
C       OUTLINE BORDER
****************************************************
C       MUST USE ALOG.LOG,@SYS:GRALIB
****************************************************
        SUBROUTINE LOG(IAOFF)
        CALL ERASE
        CALL MOVEA(0.0,0.0)
        CALL DRAWA(0.0,750.0)
        CALL DRAWA(1000.0,750.0)
        CALL DRAWA(1000.0,0.0)
        CALL DRAWA(0.0,0.0)
        CALL MOVEA(900.0,5.0)
        CALL DRAWA(900.0,750.0)
        DO 200 N=0,4
        DO 100 M=1,10
        DN=10**N
        XM=M*DN
        IF(XM.GT.35000) GO TO 225
        XX=(ALOG10(XM))*180
        XTT=900-XX
        CALL MOVEA(XTT,175.0)
        CALL DRAWR(0.0,575.0)
100     CONTINUE
200     CONTINUE
225     CALL MOVEA(900.0,175.0)
        CALL DRAWA(0.0,175.0)
        DO 400 N=0,2
        DO 300 M=1,10
        DN=10**N
        XM=M*DN
```

```
        XX=(ALOG10(XM))*190
        XT=XX+175
        CALL MOVEA(95.0,XT)
        CALL DRAWR(805.0,0.0)
300     CONTINUE
400     CONTINUE
        DO 460 M=0,3
        XN=175+190*M
        CALL MOVEA(900.0,XN)
        CALL NUMBER(10,'I2')
        CALL MOVER(0.0,10.0)
        J=IAOFF+M
        CALL NUMBER(J,'I1')
460     CONTINUE
        CALL CHRROT(90)
        CALL MOVEA(960.0,400.0)
        CALL ANCHOS('CURRE',5)
        CALL ANCHOS('NT I',4)
        CALL ANCHOS('N AMP',5)
        CALL ANCHOS('S',1)
        CALL MOVEA(900.0,0.0)
        CALL DRAWA(900.0,750.0)
        DO 500 M=0,5
        XN=900-180*M
        CALL MOVEA(XN,150.0)
        IF(M.EQ.0)CALL NUMBER(.01,'F3.2')
        IF(M.EQ.1)CALL NUMBER(.1,'F2.1')
        IF(M.EQ.2)CALL NUMBER(1,'I1')
        IF(M.EQ.3)CALL NUMBER(10,'I2')
        IF(M.EQ.4)CALL NUMBER(100,'I3')
        IF(M.EQ.5)GO TO 510
500     CONTINUE
510     CALL MOVEA(20.0,140.0)
        CALL NUMBER(1000,'I4')
        CALL CHRROT(0)
        CALL MOVEA(200.0,755.0)
        CALL ANCHOS('TIME ',5)
        CALL ANCHOS('IN SE',5)
        CALL ANCHOS('CONDS',5)
        CALL MOVEA(0.0,0.0)
        RETURN
        END
*************************************************
C       SUBROUTINE CKTBKR;COMPUTES TIME VALUE FOR A GIVEN
CURRENT
C                VALUE
************************************************
        SUBROUTINE CKTBKR(ID,MDEV,DVAL,OFSET,TVAL,AMVAL)
        CVAL=DVAL-OFSET
```

```
        IF(ID.NE.1)GO TO 50
        CVAL=CVAL+.1122
        IF(CVAL.GT.AMVAL.AND.AMVAL.NE.0)GO TO 90
        IF(CVAL.LT.0)GO TO 80
        TVAL=5.76908+(-10.88261*(CVAL))+(23.65419*(CVAL**2))+
(-34.7922
        1*(CVAL**3))+(33.77472*(CVAL**4))+(-21.14536*(CVAL**5
))+(6.4
        231778*(CVAL**6))
        TVAL=TVAL-2.0
        GO TO 100
50      CVAL=CVAL+.232
        IF(CVAL.LT.0)GO TO 80
        AMVAL=AMVAL+.075
        IF(CVAL.GT.AMVAL.AND.AMVAL.NE.0)GO TO 95
        TVAL=5.8596722+(-4.394335*CVAL)+(-.50690284*(CVAL**2)
)+(10.40
        12739*(CVAL**3))+(-17.064956*(CVAL**4))+(13.011543*(C
VAL**5))
        2+(-4.010724*(CVAL**6))
        TVAL=TVAL-2.0
        GO TO 100
80      TVAL=6.0
        GO TO 150
90      TVAL=-2.0
        GO TO 100
95      TVAL=ALOG10(.025)
100     CONTINUE
150     RETURN
        END
**********************************************
```

APPENDIX C

DISK-FILE FORMATS

Disk file RES.DAT

a_o Double precision wire resistance coefficients

.

.

a_6 Double precision wire resistance coefficients

a_o Double precision wire reactance coefficients

.

.

a_6 Double precision wire reactance coefficients

Disk file TEST.DAT

element number/line bus/load bus/positive seq p.u. impedance/zero seq p.u. impedance/transformer neutral imp line bus/transformer neutral imp load bus

00000

Load data

bus number/p.u. VA/per unit subtransient reactance

0000

Disk file XFER.DAT

line bus/load bus/inrush/low withstand/high withstand/transformer size

LCUR.DAT

line bus/load bus/current at line bus/current at load bus

FCUR.DAT

fault bus/element number/3 phase fault/1 phase fault

DEV.DAT

device number/a_o/a_1/a_2/a_3/a_4

ADEV.DAT

element number/element bus numbers/device numbers/line bus element low pickup, high pickup/load bus element low pickup, high pickup/ magnetic settings line, load/time dial setting line, load/CT ratios line,load/

APPENDIX D

SAMPLE PROGRAM DATA INPUT

Before beginning the program sequence, RES.DAT and DEV.DAT contain the proper coefficients listed in Table 8.5. All source programs have been loaded and saved. PLOTD.F4 uses subroutines contained in the PDP10 system files so it must be loaded using:

```
LOAD PLOTD.F4, @ SYS:GRALIB.
```

The first program is started with run INPUT. Underlined words and numbers are data input by the user. Words not underlined are prompts from the computer in the input sequence that follows.

```
Do you want to read from a KYBRD?
<CR> for input from TTY, 1 for read. . .
<CR>

Enter the problem base in MVA
1

Enter the problem base in KV
15

Enter element data as follows:  Bus ( ) to ( )
(Length)(Size)(Transformer impedance (P.U.))(Transformer size
in MVA)(Transformer High Voltage in KV)(Transformer low
voltage in KV)
1 2 3000 6 -(element with no transformer)
2 3 356 .001 .005 1 15 4.1 -(element with a transformer)
Input transformer ZN in P.U. refer to BUS

Side, (BUS 2) (BUS 3). . .
.001 .005 .001 .005

         (This is continued for each element until all elements of
         the type in RES.DAT have been entered)

0 -(tells computer no more elements)

         At this point if another wire type is to be used the
         following procedure applies if not skip this section
         and proceed to ①
```

```
Input Bus 1 voltage. . .0
Data is accepted and a list follows

       (Here the computer types all computed information on the
       elements entered so that they may be checked for error.)

Now enter the load data in the following manner;. . .
(Bus)(Load in KVA)(Pwr Factor). . .or. . .(Bus)(Load in HP)
(Pwr Factor). . .or. . .(Bus)(Load in Amps)(Pwr Factor)
What method will you use?. . .VA, HP, or Amps?
S -(writes element data computed to disk)
End of execution
(Load RES.DAT with the new wire type coefficients)
RUN INPUT
Do you want to read from a KYBRD?
<CR> for input from a TTY, 1 for read. . .  .
1
Old data is entered ready for new line data
Enter element data as follows. . .(Program runs as
       in the first section)

①      Picking up the program from completion of element data,
       the data entry follows.

Input Bus 1 Voltage. . .15000
Input Bus 2 Voltage. . .15000
Input Bus 3 Voltage. . .4100
      .
      .
Input Bus 8 Voltage. . .440
Data is accepted and a list follows
       (Computer types computed element data).

Number of buses is 8
Now enter the load data in the following manner;. . .
(Bus)(Load in KVA)(Pwr Factor). . .or. . .(Bus)(Load in HP)
(Pwr Factor). . .or. . .(Bus)(Load in Amps)(Pwr factor)
What method will you use?  KVA, HP, or Amps?
HP
Enter. . .
4 800 .80
What method will you use?. . .KVA, HP, or Amps?
HP
8 500 .90
       (This is continued until all loads are entered?
       What  method will you use?. . .KVA, HP, or Amps?
S
End of Execution
RUN LDFLO
End of Execution
RUN ZBUS
Enter the zero sequence impedance for the loads at each bus. . .
```

```
8 .001 .005 -for each load where the zero sequence impedance
     :        will not be assumed to equal the positive sequence
     :        impedance
0 -tells the computer there are no more entries
Enter the mutually coupled elements in per unit in the following
format, TEST.DAT has the element numbers. . .,(Element)(Element)
(Zero Sequence Value)
2 3 0 .001 (The zero must be entered because the coupling is a
reactive value)
0 -(no more or no coupling)
End of execution
RUN FALT
Enter the fault impedance (Real)(Imaginary)
0 0
End of execution
RUN COORD
All load currents are ready, use device number list
Select device for the load at bus 4
4
.
:
Select the device for the load at Bus 8
3
Input temperature for ambient shift, 1 for less than
110 Deg C, 2 for 60 Deg C
<CR> for no shift. . .
<CR>
Does the device have a magnetic element?(0)Yes
0
Select device for Bus 6 protecting element feeding load at Bus 8
4
Select the Westinghouse Co. relay, refer to Table 1
Does the relay have an inst. element if yes 0
0
.
:
       (Until all backup devices have been set).
       Enter the device for the Bus 6 of element 5
4
Select the Westinghouse Co. relay, refer to Table 1
Does the relay have an inst. element if yes 0
0
.
:
       (Until all devices in the system have been set).
Device listing -(Here the computer lists the matrix ADEV)
End of execution
Run PLOTD
Devices have been set if you want to plot graph paper 0 if not
type 1
0
```

```
This will plot curves based on the lowest voltage in the
system
Input the current offset for the graph, integer 1
Input element number
7
Finished?
1 -(no, 0 yes)
Devices have been set if you want to plot graph paper
0 if not type 1
1
Input element number
6
Finished?
0
End of execution.
```

If a device did not plot as desired its settings can be changed in ADEV.DAT.

202479
Thesis
S1545 Salmond
C.1 A computational
procedure for the
protection of ind-
ustrial power systems.

202479
Thesis
S1545 Salmond
C.1 A computational
procedure for the
protection of ind-
ustrial power systems.

thesS1545missing
A computational procedure for the protec
3 2768 001 00147 2
DUDLEY KNOX LIBRARY